

ModelArts

User Guide (Standard)

Issue	01
Date	2025-08-29



Copyright © Huawei Cloud Computing Technologies Co., Ltd. 2025. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Cloud Computing Technologies Co., Ltd.

Trademarks and Permissions



HUAWEI and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei Cloud and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Cloud Computing Technologies Co., Ltd.

Address: Huawei Cloud Data Center Jiaoxinggong Road
Qianzhong Avenue
Gui'an New District
Gui Zhou 550029
People's Republic of China

Website: <https://www.huaweicloud.com/intl/en-us/>

Contents

1 ModelArts Standard Usage.....	1
2 ModelArts Standard Preparations.....	5
2.1 Configuring Access Authorization for ModelArts Standard.....	5
2.1.1 Configuring Agency Authorization for ModelArts with One Click.....	5
2.1.2 Creating an IAM User and Granting ModelArts Permissions.....	11
2.2 Creating and Managing a Workspace.....	19
2.3 Creating an OBS Bucket for ModelArts to Store Data.....	24
3 ModelArts Standard Resource Management.....	27
3.1 About ModelArts Standard Resource Pools.....	27
3.2 Creating a Standard Dedicated Resource Pool.....	29
3.3 Managing Standard Dedicated Resource Pools.....	43
3.3.1 Viewing Details About a Standard Dedicated Resource Pool.....	43
3.3.2 Resizing a Standard Dedicated Resource Pool.....	51
3.3.3 Upgrading the Standard Dedicated Resource Pool Driver.....	55
3.3.4 Rectifying a Faulty Node in a Standard Dedicated Resource Pool.....	59
3.3.5 Modifying the Job Types Supported by a Standard Dedicated Resource Pool.....	69
3.3.6 Migrating Standard Dedicated Resource Pools and Networks to Other Workspaces.....	70
3.3.7 Configuring the Standard Dedicated Resource Pool to Access the Internet.....	72
3.3.8 Using TMS Tags to Manage Resources by Group.....	79
3.3.9 Managing Logical Subpools of a Standard Dedicated Resource Pool.....	81
3.3.10 Releasing Standard Dedicated Resource Pools and Deleting the Network.....	87
3.4 Managing Standard Dedicated Resource Pool Plug-ins.....	90
3.4.1 Overview.....	90
3.4.2 Node Fault Detection (ModelArts Node Agent).....	99
3.4.3 ModelArts Metric Collector.....	99
3.4.4 AI Suite (NV GPU).....	101
3.4.5 AI Suite (Modelarts Device Plugin).....	104
3.4.6 Volcano Scheduler.....	105
3.4.7 NodeLocal DNSCache.....	107
3.4.8 Cloud Native Log Collection.....	112
3.4.9 kube-prometheus-stack.....	115
4 Using Workflows for Low-Code AI Development.....	121

4.1 What Is Workflow?	121
4.2 Managing a Workflow	124
4.2.1 Searching for a Workflow	125
4.2.2 Viewing the Running Records of a Workflow	126
4.2.3 Managing a Workflow	128
4.2.4 Retrying, Stopping, or Running a Workflow Phase	129
4.3 Workflow Development Command Reference	130
4.3.1 Core Concepts of Workflow Development	130
4.3.2 Configuring Workflow Parameters	138
4.3.3 Configuring the Input and Output Paths of a Workflow	140
4.3.4 Creating Workflow Phases	144
4.3.4.1 Creating a Dataset Phase	144
4.3.4.2 Creating a Dataset Labeling Phase	150
4.3.4.3 Creating a Dataset Import Phase	156
4.3.4.4 Creating a Dataset Release Phase	165
4.3.4.5 Creating a Training Job Phase	171
4.3.4.6 Creating a Model Registration Phase	188
4.3.4.7 Creating a Service Deployment Phase	196
4.3.5 Creating a Multi-Branch Workflow	204
4.3.5.1 Multi-Branch Workflow	204
4.3.5.2 Creating a Condition Phase to Control Branch Execution	204
4.3.5.3 Configuring Phase Parameters to Control Branch Execution	211
4.3.5.4 Configuring Multi-Branch Phase Data	216
4.3.6 Creating a Workflow	218
4.3.7 Publishing a Workflow	220
4.3.7.1 Publishing a Workflow to ModelArts	220
4.3.7.2 Publishing a Workflow to AI Gallery	222
4.3.8 Advanced Workflow Capabilities	224
4.3.8.1 Using Big Data Capabilities (MRS) in a Workflow	224
4.3.8.2 Specifying Certain Phases to Run in a Workflow	225
5 Using Notebook for AI Development and Debugging	227
5.1 Application Scenarios	227
5.2 Creating a Notebook Instance (Default Page)	228
5.3 Creating a Notebook Instance (New Page)	243
5.4 Managing Notebook Instances	258
5.4.1 Searching for a Notebook Instance	258
5.4.2 Updating a Notebook Instance	259
5.4.3 Starting, Stopping, or Deleting a Notebook Instance	262
5.4.4 Saving a Notebook Instance	262
5.4.5 Dynamically Expanding EVS Disk Capacity	265
5.4.6 Dynamically Mounting an OBS Parallel File System	266
5.4.7 Viewing Notebook Events	268

5.4.8 Notebook Cache Directory Alarm Reporting.....	273
5.5 Using a Notebook Instance for AI Development Through JupyterLab.....	277
5.5.1 Using JupyterLab to Develop and Debug Code Online.....	277
5.5.2 Common Functions of JupyterLab.....	279
5.5.3 Using Git to Clone the Code Repository in JupyterLab.....	290
5.5.4 Creating a Scheduled Job in JupyterLab.....	296
5.5.5 Uploading Files to JupyterLab.....	300
5.5.5.1 Uploading Files from a Local Path to JupyterLab.....	300
5.5.5.2 Cloning GitHub Open-Source Repository Files to JupyterLab.....	307
5.5.5.3 Uploading OBS Files to JupyterLab.....	308
5.5.5.4 Uploading Remote Files to JupyterLab.....	311
5.5.6 Downloading a File from JupyterLab to a Local PC.....	312
5.5.7 Using MindInsight Visualization Jobs in JupyterLab.....	315
5.5.8 Using TensorBoard Visualization Jobs in JupyterLab.....	317
5.6 Using Notebook Instances Remotely Through PyCharm.....	320
5.6.1 Connecting to a Notebook Instance Through PyCharm Toolkit.....	321
5.6.2 Manually Connecting to a Notebook Instance Through PyCharm.....	329
5.6.3 Uploading Data to a Notebook Instance Through PyCharm.....	335
5.7 Using Notebook Instances Remotely Through VS Code.....	336
5.7.1 Connecting to a Notebook Instance Through VS Code.....	336
5.7.2 Connecting to a Notebook Instance Through VS Code Toolkit.....	337
5.7.3 Manually Connecting to a Notebook Instance Through VS Code.....	346
5.7.4 Uploading and Downloading Files in VS Code.....	350
5.8 Using a Notebook Instance Remotely with SSH	352
5.9 ModelArts CLI Command Reference.....	358
5.9.1 ModelArts CLI Commands.....	358
5.9.2 (Optional) Installing ma-cli Locally.....	359
5.9.3 Autocompletion for ma-cli Commands.....	360
5.9.4 ma-cli Authentication.....	361
5.9.5 ma-cli image Commands for Building Images.....	364
5.9.6 ma-cli ma-job Commands for Training Jobs.....	375
5.9.7 ma-cli dli-job Commands for Submitting DLI Spark Jobs.....	387
5.9.8 Using ma-cli to Copy OBS Data.....	400
5.10 Using MoXing Commands in a Notebook Instance.....	401
5.10.1 MoXing Framework Functions.....	401
5.10.2 Using MoXing in Notebook.....	403
5.10.3 Introducing MoXing Framework.....	405
5.10.4 Mapping Between mox.file and Local APIs and Switchover.....	406
5.10.5 Sample Code for Common Operations.....	407
5.10.6 Sample Code for Advanced MoXing Usage.....	411
6 Preparing and Processing Data.....	414
6.1 Preparing Data.....	414

6.2 Creating a ModelArts Dataset.....	415
6.3 Importing Data to a ModelArts Dataset.....	432
6.3.1 Introduction to Data Importing.....	432
6.3.2 Importing Data from OBS.....	433
6.3.2.1 Introduction to Importing Data from OBS.....	433
6.3.2.2 Importing Data from an OBS Path to ModelArts.....	436
6.3.2.3 Specifications for Importing Data from an OBS Directory.....	438
6.3.2.4 Importing a Manifest File to ModelArts.....	444
6.3.2.5 Specifications for Importing a Manifest File.....	445
6.3.3 Importing Data from MRS to ModelArts.....	463
6.3.4 Importing Data from Local Files.....	463
6.4 Labeling ModelArts Data.....	464
6.4.1 Scenarios.....	465
6.4.2 Manual Labeling.....	467
6.4.2.1 Creating a Manual Labeling Job.....	467
6.4.2.2 Labeling Images.....	474
6.4.2.3 Labeling Text.....	483
6.4.2.4 Labeling Audio.....	487
6.4.2.5 Labeling Video.....	490
6.4.2.6 Managing Labeling Data.....	491
6.4.3 Auto Labeling.....	496
6.4.3.1 Creating an Auto Labeling Job.....	496
6.4.3.2 Hard Examples of an Auto Labeling Job.....	500
6.4.3.3 Auto Grouping for Labeling Jobs.....	502
6.4.4 Team Labeling.....	504
6.4.4.1 Using Team Labeling.....	504
6.4.4.2 Creating and Managing Teams.....	504
6.4.4.3 Creating a Team Labeling Job.....	505
6.4.4.4 Reviewing and Accepting Team Labeling Results.....	510
6.4.4.5 Managing Teams and Team Members.....	514
6.4.5 Managing Labeling Jobs.....	515
6.5 Publishing ModelArts Data.....	517
6.6 Analyzing ModelArts Data Characteristics.....	520
6.7 Exporting Data from a ModelArts Dataset.....	525
6.7.1 Exporting Data from ModelArts to OBS.....	525
6.7.2 Exporting Data as a New Dataset.....	526
6.8 Getting Started: Creating an Object Detection Dataset.....	527
7 Using ModelArts Standard to Train Models.....	535
7.1 Model Training Process.....	535
7.2 Preparing Model Training Code.....	538
7.2.1 Starting Training Using a Preset Image's Boot File.....	538
7.2.2 Developing Code for Training Using a Preset Image.....	547

7.2.3 Developing Code for Training Using a Custom Image.....	550
7.2.4 Configuring Password-free SSH Mutual Trust Between Instances for a Training Job Created Using a Custom Image.....	553
7.3 Preparing a Model Training Image.....	554
7.4 Creating an Algorithm.....	556
7.5 Creating a Production Training Job.....	567
7.6 Creating a Production Training Job (New Version).....	585
7.7 Distributed Model Training.....	603
7.7.1 Overview.....	603
7.7.2 Creating a Single-Node Multi-PU Distributed Training Job (DataParallel).....	605
7.7.3 Creating a Multiple-Node Multi-PU Distributed Training Job (DistributedDataParallel).....	606
7.7.4 Example: Creating a DDP Distributed Training Job (PyTorch + GPU).....	617
7.7.5 Example: Creating a DDP Distributed Training Job (PyTorch + NPU).....	621
7.8 Enabling Dynamic Route Acceleration for Training Jobs.....	622
7.9 Incremental Model Training.....	625
7.10 Automatic Model Tuning (AutoSearch).....	627
7.10.1 Overview.....	628
7.10.2 Creating a Training Job for Automatic Model Tuning.....	629
7.11 High Model Training Reliability.....	632
7.11.1 Training Job Fault Tolerance Check.....	632
7.11.2 Training Log Failure Analysis.....	638
7.11.3 Detecting Training Job Suspension.....	639
7.11.4 Training Job Restart Upon Suspension.....	642
7.11.5 Resumable Training.....	645
7.11.6 Enabling Unconditional Auto Restart.....	647
7.12 Configuring Supernode Affinity Group Instances.....	649
7.13 Managing Model Training Jobs.....	651
7.13.1 Viewing Training Job Details.....	651
7.13.2 Visualizing the Training Job Process.....	656
7.13.3 Viewing the Resource Usage of a Training Job.....	657
7.13.4 Viewing the Model Evaluation Result.....	661
7.13.5 Viewing Training Job Events.....	665
7.13.6 Viewing Training Job Logs.....	667
7.13.7 Priority of a Training Job.....	675
7.13.8 Using Cloud Shell to Debug a Production Training Job.....	676
7.13.9 Saving the Image of a Debug Training Job.....	681
7.13.10 Copying, Stopping, or Deleting a Training Job.....	682
7.13.11 Managing Environment Variables of a Training Container.....	683
7.13.12 Viewing Training Job Tags.....	690
7.13.13 Managing Training Experiments.....	690
7.13.14 Viewing Monitoring Metrics of a Training Job.....	693
8 Using ModelArts Standard to Deploy Models for Inference and Prediction.....	697

8.1 Overview.....	697
8.2 Creating a Model.....	700
8.2.1 Creation Methods.....	700
8.2.2 Importing a Meta Model from a Training Job.....	702
8.2.3 Importing a Meta Model from OBS.....	704
8.2.4 Importing a Meta Model from a Container Image.....	709
8.3 Model Creation Specifications.....	713
8.3.1 Model Package Structure.....	714
8.3.2 Specifications for Editing a Model Configuration File.....	715
8.3.3 Specifications for Writing a Model Inference Code File.....	731
8.3.4 Specifications for Using a Custom Engine to Create a Model.....	736
8.3.5 Examples of Custom Scripts.....	739
8.4 Deploying a Model as Real-Time Inference Jobs.....	750
8.4.1 Deploying and Using Real-Time Inference.....	750
8.4.2 Deploying a Model as a Real-Time Service.....	751
8.4.3 Authentication Methods for Accessing Real-time Services.....	762
8.4.3.1 Accessing a Real-Time Service Through Token-based Authentication.....	763
8.4.3.2 Accessing a Real-Time Service Through AK/SK-based Authentication.....	771
8.4.3.3 Accessing a Real-Time Service Through App Authentication.....	777
8.4.4 Accessing a Real-Time Service Through Different Channels.....	788
8.4.4.1 Accessing a Real-Time Service Through a Public Network.....	788
8.4.4.2 Accessing a Real-Time Service Through a VPC Channel.....	789
8.4.4.3 Accessing a Real-Time Service Through a VPC High-Speed Channel.....	791
8.4.5 Accessing a Real-Time Service Using Different Protocols.....	795
8.4.5.1 Accessing a Real-Time Service Using WebSocket.....	795
8.4.5.2 Accessing a Real-Time Service Using Server-Sent Events.....	798
8.5 Deploying a Model as a Batch Inference Service.....	800
8.6 Managing ModelArts Models.....	807
8.6.1 Viewing ModelArts Model Details.....	807
8.6.2 Viewing ModelArts Model Events.....	812
8.6.3 Managing ModelArts Model Versions.....	816
8.7 Managing a Synchronous Real-Time Service.....	817
8.7.1 Viewing Details About a Real-Time Service.....	817
8.7.2 Viewing Events of a Real-Time Service.....	824
8.7.3 Managing the Lifecycle of a Real-Time Service.....	826
8.7.4 Modifying a Real-Time Service.....	828
8.7.5 Viewing Performance Metrics of a Real-Time Service on Cloud Eye.....	829
8.7.6 Integrating a Real-Time Service API into the Production Environment.....	836
8.7.7 Configuring Auto Restart upon a Real-Time Service Fault.....	837
8.8 Managing Batch Inference Jobs.....	837
8.8.1 Viewing Details About a Batch Service.....	837
8.8.2 Viewing Events of a Batch Service.....	839

8.8.3 Managing the Lifecycle of a Batch Service.....	842
8.8.4 Modifying a Batch Service.....	843
9 Creating a Custom Image for ModelArts Standard.....	845
9.1 Applications of Custom Images.....	845
9.2 Preset Images Supported by ModelArts.....	847
9.2.1 ModelArts Preset Image Updates.....	847
9.2.2 ModelArts Unified Images.....	849
9.2.3 Preset Dedicated Images in Notebook Instances.....	867
9.2.4 Preset Dedicated Images for Training.....	895
9.2.5 Preset Dedicated Images for Inference.....	898
9.3 Creating a Custom Image for a Notebook Instance.....	914
9.3.1 Creating a Custom Image.....	914
9.3.2 Creating a Custom Image on ECS and Using It.....	916
9.3.3 Creating a Custom Image Using Dockerfile.....	922
9.3.4 Creating a Custom Image Using the Image Saving Function.....	925
9.4 Creating a Custom Image for Model Training.....	927
9.4.1 Creating a Custom Training Image.....	928
9.4.2 Creating a Custom Training Image Using a Preset Image.....	929
9.4.3 Migrating Existing Images to ModelArts.....	932
9.4.4 Creating a Custom Training Image (PyTorch + Ascend).....	934
9.4.5 Creating a Custom Training Image (PyTorch + CPU/GPU).....	940
9.4.6 Creating a Custom Training Image (MPI + CPU/GPU).....	945
9.4.7 Creating a Custom Training Image (Tensorflow + GPU).....	953
9.4.8 Creating a Custom Training Image (MindSpore + Ascend).....	960
9.5 Creating a Custom Image for Inference.....	977
9.5.1 Creating a Custom Image for a Model.....	977
9.5.2 Creating a Custom Image on ECS.....	979
10 Monitoring ModelArts Standard Resources.....	984
10.1 Overview.....	984
10.2 Viewing Monitoring Metrics on the ModelArts Console.....	985
10.3 Viewing All ModelArts Monitoring Metrics on the AOM Console.....	986
10.4 Using Grafana to View AOM Monitoring Metrics.....	1048
10.4.1 Installing and Configuring Grafana.....	1048
10.4.1.1 Installing and Configuring Grafana on Windows.....	1048
10.4.1.2 Installing and Configuring Grafana on Linux.....	1049
10.4.1.3 Installing and Configuring Grafana on a Notebook Instance.....	1052
10.4.2 Configuring a Grafana Data Source.....	1055
10.4.3 Configuring a Dashboard to View Metric Data.....	1059
11 Using CTS to Audit ModelArts Standard.....	1065
11.1 ModelArts Standard Key Operations Traced by CTS.....	1065
11.2 Viewing ModelArts Standard Audit Logs.....	1073

1 ModelArts Standard Usage

This chapter aims to help you learn how to use ModelArts Standard and get started with the ModelArts service quickly.

For developers who are experienced in coding, debugging, and working with AI engines, ModelArts provides online coding environments as well as an E2E AI development process that covers data preparation, model training, model management, and service deployment.

This document describes how to perform AI development on the ModelArts management console. If you use the APIs or SDKs for development, view [ModelArts SDK Reference](#) or [ModelArts API Reference](#).

To view the examples of AI development lifecycle, see [Getting Started](#) and [Best Practices](#).

Application Scenarios of ModelArts Standard

ModelArts Standard is a one-stop development platform for AI developers. It provides a user-friendly management console with end-to-end AI development toolchains, covering data management, development environment, model training, model management, and service deployment.

- ModelArts Standard's workflow is a low-code AI development pipeline tool, covering data labeling, data processing, model development, training, model evaluation, and service deployment. Workflows are executed in visualized mode. For details, see [What Is Workflow?](#).
- ModelArts Standard's development environment, notebook, provides a cloud-based JupyterLab environment and local IDE plug-ins, helping you write training and inference code and use cloud resources to debug the code. For details, see [Notebook Application Scenarios](#).
- ModelArts Standard's model training provides GUI-based training, debugging, and production environments. You can use your own data and algorithms to train models using the compute resources provided by ModelArts Standard. For details, see [Using ModelArts Standard to Train Models](#).
- ModelArts Standard's inference deployment provides a GUI-based production environment for inference deployment. After an AI model is developed, you can manage it and quickly deploy it as an inference service. You can perform online inference and prediction or integrate AI inference capabilities into your IT platform by calling APIs. For details, see [Overview](#).

Process for Using ModelArts Standard

The AI development lifecycle on ModelArts Standard allows you to experience end-to-end AI development, from preparing data to deploying a model as a service. It takes developers' habits into consideration and provides a variety of engines and scenarios for you to choose. You can use the ModelArts Standard functions as needed in each phase during AI development. The following describes the entire process from data preparation to service development using ModelArts.

Figure 1-1 ModelArts Standard usage process

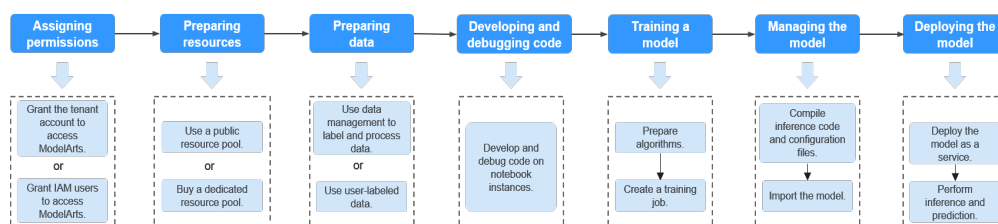


Table 1-1 Process description

Task	Subtask	Description	Reference
Assigning permissions	Configuring agency authorization for ModelArts	ModelArts depends on other cloud services, and you need to configure agency authorization to allow ModelArts to access these services.	Configuring Agency Authorization for ModelArts with One Click
(Optional) Creating an OBS bucket	Creating an OBS bucket for ModelArts to store data	ModelArts does not support data storage itself. The input data, output data, and cached data generated during AI development using ModelArts Standard can be stored in OBS buckets. Therefore, you are advised to create an OBS bucket before using ModelArts. You can also create an OBS bucket later when needed.	Creating an OBS Bucket for ModelArts to Store Data

Task	Subtask	Description	Reference
(Optional) Preparing resources	Creating a dedicated ModelArts Standard resource pool	ModelArts Standard supports both public and dedicated resource pools. Public resource pool: When creating a training or inference task, you can choose the public resource pool directly without having to create one by yourself. If you use the public resource pool, skip this step. Dedicated resource pool: You need to purchase and create a dedicated resource pool first, but the resources are exclusively used by yourself. This step is mandatory if you use the dedicated resource pool.	Creating a Standard Dedicated Resource Pool
(Optional) Preparing data	Creating a dataset	ModelArts Standard supports data management. You can create datasets in ModelArts Standard for managing, preprocessing, and labeling data. If you have prepared data for training, you can directly upload the data to OBS without using the data management function.	Creating a Dataset Labeling Data Publishing a Dataset
Developing and debugging code in the development environment	Creating a notebook instance	Create a notebook instance as the development environment for debugging training and inference code. You are advised to debug the training code in the development environment before creating a production training job.	Creating a Notebook Instance

Task	Subtask	Description	Reference
Training a model	Preparing algorithms	Before creating a training job, you need to prepare an algorithm. You can subscribe to an algorithm in AI Gallery or use your own algorithm.	Preparing Algorithms
	Creating a training job	Create a training job, select the available dataset version, and use the compiled training script. After training is complete, a model is generated and stored in OBS.	Creating a Training Job
Managing models	Compiling inference code and configuration files	Following the model package specifications provided by ModelArts, compile inference code and configuration files for your model, and save them to the training output path.	Model Package Specifications
	Creating a model	Import a trained model to ModelArts to create a model, facilitating model deployment and publishing.	Creating a Model
Deploying models	Deploying a model as a service	Deploy a model as a real-time, batch, or edge service.	• Deploying a Model as a Real-Time Service • Deploying a Model as a Batch Service
	Accessing the service	After the service is deployed, access the real-time or edge service, or view the prediction result of the batch service.	• Accessing a Real-Time Service • Viewing the Prediction Result of a Batch Service

2 ModelArts Standard Preparations

2.1 Configuring Access Authorization for ModelArts Standard

2.1.1 Configuring Agency Authorization for ModelArts with One Click

Scenarios

ModelArts, the AI platform, must access other services when executing tasks. For example, ModelArts must access OBS to read your data for training. In such cases, ModelArts accesses other cloud services on behalf of you. To ensure security, ModelArts requires your authorization before accessing any cloud services, which is the agency process. Then, you can perform AI computing tasks on ModelArts.

ModelArts allows you to add authorization. You can quickly configure agency authorization on the **Permission Management** page of ModelArts Standard. Then, ModelArts will automatically create an agency for you and configure it in ModelArts.

ModelArts Standard supports common and high-privilege permissions.

- **Common:** In this mode, users can customize the authorization scope based on service scenarios and grant only the minimum required access. This mode is flexible and secure. This mode applies to scenarios with strict requirements on the authorization scope.
- **High-privilege:** The configuration of this mode is simple, but the agency created has high permissions. This mode is suitable for users who require administrator permissions. If you need to precisely control the permissions granted to an agency, the common permissions are recommended.

Constraints

- Huawei Cloud account
 - Only a Huawei Cloud account can use an agency to authorize the current account or all IAM users under the current account.

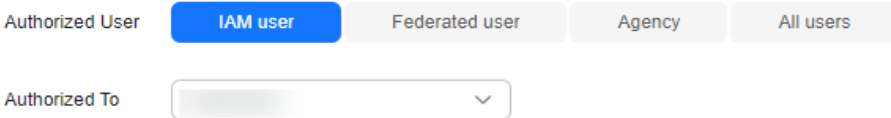
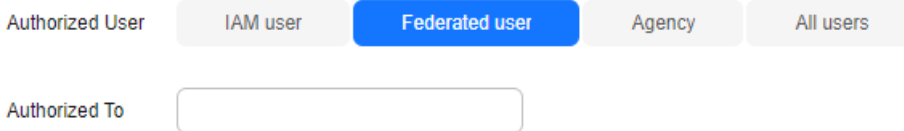
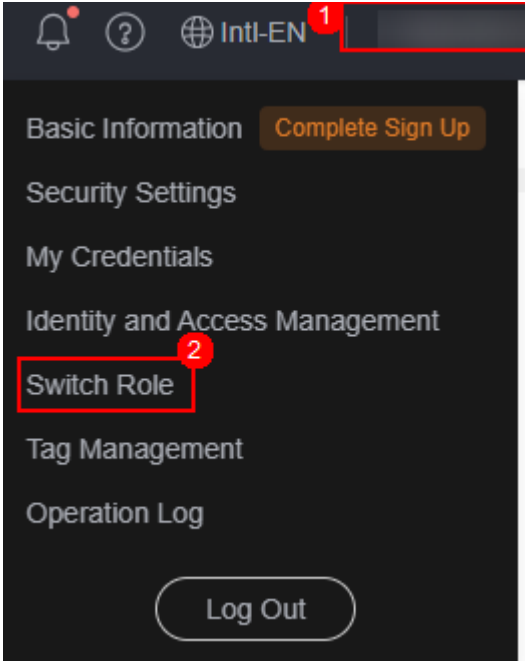
- Multiple IAM users or accounts can use the same agency.
- A maximum of 100 agencies can be created under an account.
- If you use ModelArts for the first time, add an agency. Generally, common user permissions are sufficient for your requirements. You can also customize permissions for refined permissions management.
- IAM user
 - If you have obtained the authorization, you can view the authorization information on the **Permission Management** page.
 - If you have not been authorized, ModelArts will display a message indicating that you have not been authorized when you access the **Add Authorization** page. In this case, contact your administrator to add authorization.

Adding Authorization

1. Log in to the ModelArts console. In the navigation pane on the left, choose **System Management > Permission Management**. The **Permission Management** page is displayed.
2. Click **Add Authorization new**. On the **Add Authorization** page that is displayed, set parameters.

Table 2-1 Parameters

Parameter	Description
Authorized User	The options are IAM user, Federated user, Agency, and All users. • IAM user: You can use a tenant account to create IAM users and assign permissions for specific resources. Each IAM user has their own identity credentials (password and access keys) and uses cloud resources based on the assigned permissions. For details about IAM users, see IAM User. • Federated user: A federated user is also called a virtual enterprise user. For details about federated users, see Configuring Federated Identity Authentication. • Agency: You can create agencies in IAM. For details about how to create an agency, see Creating an Agency. • All users: If you select this option, the agency permissions will be granted to all IAM users under the account, including those created in the future. For individual users, select All users.

Parameter	Description
Authorized To	This parameter is not displayed when Authorized User is set to All users. IAM user: Select an IAM user and configure an agency for the IAM user. Figure 2-1 Selecting an IAM user  Federated user: Enter the username or user ID of the target federated user. Figure 2-2 Entering a federated user  Agency: Select an agency name. You can create an agency under account A and grant the agency permissions to account B. When using account B, you can switch the role in the upper right corner of the console to account A and use the agency permissions of account A. Figure 2-3 Switch Role 

Parameter	Description
Agency	● Use existing: If there are agencies in the list, select an available one to authorize the selected user. Click the drop-down arrow next to an agency name to view its permission details. ● Add agency: If there is no available agency, create one. If you use ModelArts for the first time, select Add agency.
Add agency > Agency Name	ModelArts automatically creates an agency name for you, but it is editable.
Add agency > Permissions > Common	In common mode, users can customize the authorization scope based on service scenarios and grant only the minimum required access. This mode is flexible and secure. This mode applies to scenarios with refined management requirements on the authorization scope. You can select the required permission from the list. The common mode provides preset permissions templates that allow for quick configuration of permissions. For example, when selecting the permission template for the training module, the system automatically chooses all necessary service and function permissions.
Add agency > Permissions > High-privilege	In high-privilege mode, higher permissions are assigned. This mode is suitable for users who require administrator permissions. To quickly find specific permissions, enter keywords into the search box. Figure 2-4 High-privilege Permissions Common Suitable for users with common permission needs. Customize authorization based on service scenarios and grant only the minimum required access for security. High-privilege Suitable for users with high permission needs. Configure permissions carefully in this mode. Select a property or enter a keyword. ☐ Policy/Role Module Description ☐ DWS Administrator Datasets Label Data Data Warehouse Service Administrator ☐ VPC Administrator Algorithm Management Training Jobs Model Management Real-Time Services Batch Services ... VPC Administrator ☐ CES ReadOnlyAccess Model Management Real-Time Services Batch Services Read-only permissions for Cloud Eye. ☐ EPS FullAccess Algorithm Management Training Jobs Model Management Real-Time Services Batch Services All operations on the Enterprise Project Mana... ☐ CTS Administrator Datasets Label Data Notebook Image Management Elastic Server Algorithm Management T... CTS Administrator ☐ SFS ReadOnlyAccess Algorithm Management Training Jobs The read-only permissions to all Scalable Fil... ☐ LTS FullAccess Model Management Real-Time Services Batch Services All permissions of Log Tank service. ☐ ECS FullAccess Notebook Image Management Elastic Server All permissions of ECS service. Create

3. Select **I have read and agree to the ModelArts Service Statement**. Click **Create**.

Viewing Authorized Permissions

You can view the configured authorizations on the **Permission Management** page. Locate an authorization and click **View Permissions** in the **Operation** column to view the permission details.

Figure 2-5 View Permissions

Authorized To	Authori...	Authori...	Authorization Content	Creation Time	Operation
All users	All users	Agency	modelarts_agency	T=0...	View Permissions Delete

Modifying the Authorization Scope

- To modify the authorization scope, click **Modify permissions in IAM** in the **View Permissions** dialog box.

Figure 2-6 Modify permissions in IAM

View Permissions	
Authorized To	All users
Agency Name	modelarts_agency
Agency Permission	15 permissions Modify permissions in IAM

- On the **Agencies** page of the IAM console, locate the target agency and modify the agency information on the **Basic Information** tab. Set **Validity Period** as needed. The value can be **Unlimited**, **1 day**, or **Custom**. If you select **Custom**, you can then enter the number of days you want the agency to stay valid for, for example, **30**.

Figure 2-7 Agency information

Agencies / modelarts_agency

Basic Information Permissions

Agency Name: modelarts_agency

URN: iam: [redacted] :agency:modelarts_agency

★ Agency Type: Cloud service

★ Cloud Service: ModelArts

★ Validity Period: Unlimited

Description: Created by ModelArts service. 29/255

OK Cancel

3. On the **Permissions** page, click **Authorize**, select policies or roles, and click **Next**. Select the scope for minimum authorization and click **OK**.

When setting the minimum authorization scope, you can select specified projects or all resources. If you select **All resources**, the selected permissions will be applied to all resources.

Deleting an Authorization

To better manage your authorization, you can delete the authorization of an IAM user or delete the authorizations of all users in batches.

- **Deleting the authorization of a user**

The authorizations configured for the IAM user of the current account are displayed on the **Permission Management** page. You can locate an authorization, click **Delete** in the **Operation** column, enter **DELETE** in the text box, and click **OK**. After it is deleted, the user cannot use ModelArts functions.

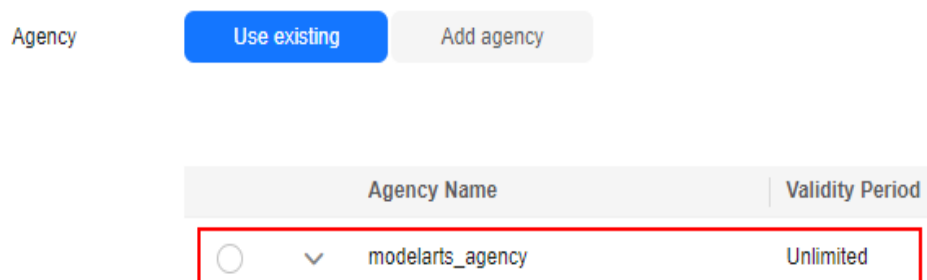
- **Deleting authorizations in batches**

On the **Permission Management** page, click **Clear Authorization** above the authorization list. Enter **DELETE** in the text box and click **OK**. After the deletion, the account and all its IAM users cannot use ModelArts functions.

FAQs

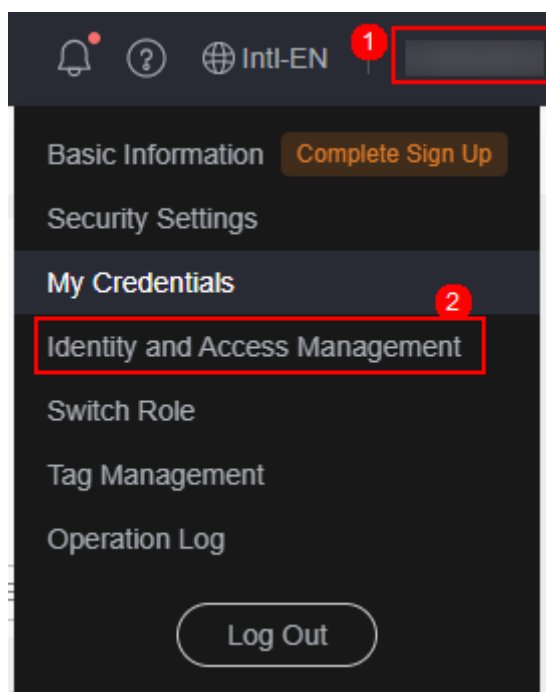
1. How do I delete an existing agency from the agency list?

Figure 2-8 Use existing



Go to the IAM console, choose **Agencies** in the navigation pane, and delete the target agency.

Figure 2-9 Identity and Access Management



2. Why is a message indicating insufficient permissions displayed when I access a page on the ModelArts console?

The permissions configured for the user agency are insufficient or the module has been upgraded. The authorization information needs to be updated. In this case, you can add authorization as prompted.

2.1.2 Creating an IAM User and Granting ModelArts Permissions

The agency created in [Configuring Agency Authorization for ModelArts with One Click](#) almost has all permissions of dependency services. If your Huawei Cloud account meets your permissions requirements, you can skip this section.

ModelArts allows you to configure fine-grained permissions for refined management of resources and permissions. This type of feature is commonly used

in scenarios with large-scale enterprise users. If you want to precisely control the scope of permissions granted to an agency, use custom authorization.

This section describes how to configure fine-grained permissions for IAM users.

Using ModelArts requires OBS authorization. ModelArts users require OBS system permissions.

- For details about how to grant operation permissions of ModelArts and dependent services to a user, see [Configuring Common Operations Permissions for ModelArts Standard](#).
- For details about how to manage user permissions on ModelArts and dependent services in a refined manner and how to configure custom policies, see [Creating a Custom Policy for ModelArts Standard](#).

Prerequisites

You have learned about the permissions that can be added to user groups for using ModelArts and dependent services so that you can select them based on your requirements. For details about the permissions supported by ModelArts, see [Table 2-2](#).

Table 2-2 Service authorizations

Target Service	Authorization Description	IAM Permission	Mandatory
ModelArts	This permission allows sub-users to use ModelArts. The sub-users with the ModelArts CommonOperations permission can only use resources, but cannot create, update, or delete any dedicated resource pool. You are advised to assign this permission to sub-users.	ModelArts CommonOperations	Yes
	The sub-users with the ModelArts FullAccess permission have all access permissions, including creating, updating, and deleting dedicated resource pools. Exercise caution when assigning this permission.	ModelArts FullAccess	No Select either ModelArts FullAccess or ModelArts CommonOperations .

Target Service	Authorization Description	IAM Permission	Mandatory
OBS	This permission allows sub-users to use OBS. ModelArts datasets, development environments, training jobs, and model inference and deployment require OBS for forwarding data .	OBS OperateAccess	Yes
SWR	This permission allows sub-users to use SWR. ModelArts custom images require the SWR FullAccess permission.	SWR OperateAccess	Yes
KMS	This permission allows sub-users to use remote SSH of ModelArts notebook.	KMS CMKFullAccess	No
IEF	This permission allows sub-users to use IEF. It is required if you use ModelArts edge services that depend on IEF.	Tenant Administrator	No
Cloud Eye	This permission allows sub-users to use Cloud Eye. Using Cloud Eye, you can view the running statuses of ModelArts real-time services and model loads, and set monitoring alarms.	CES FullAccess	No
SMN	This permission allows sub-users to use SMN. SMN is used with Cloud Eye.	SMN FullAccess	No
VPC	When creating a ModelArts dedicated resource pool, sub-users require VPC permissions so that they can customize networks.	VPC FullAccess	No

Target Service	Authorization Description	IAM Permission	Mandatory
SFS	This permission allows sub-users to use SFS. SFS file systems can be mounted to ModelArts dedicated resource pools to serve as storage for development environments or training jobs.	SFS Turbo FullAccess SFS FullAccess	No

Configuring Common Operations Permissions for ModelArts Standard

Step 1 Create a user group.

[Log in to the IAM console](#). Choose **User Groups** and click **Create User Group**. Enter a user group name and click **OK**.

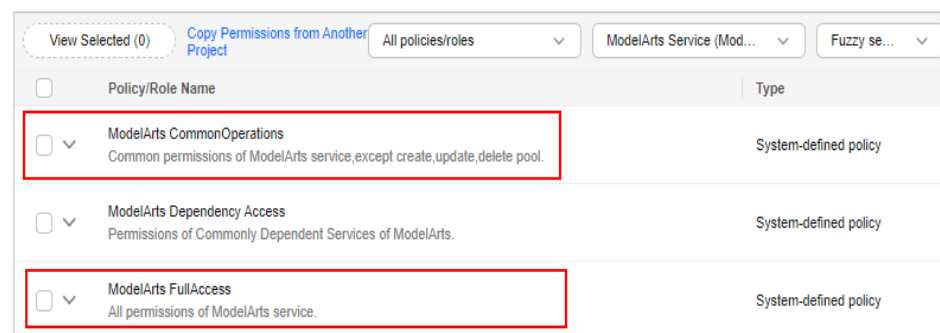
Step 2 Configure permissions for the user group.

In the user group list, locate the created user group, click **Authorize** in the **Operation** column, and perform the following operations.

- Assign permissions for using ModelArts. Search for **ModelArts** in the search box. Select either **ModelArts FullAccess** or **ModelArts CommonOperations**. The differences between the options are as follows:
 - The sub-users with the **ModelArts CommonOperations** permission can only use resources, but cannot create, update, or delete any dedicated resource pool. You are advised to assign this permission to sub-users.
 - The sub-users with the **ModelArts FullAccess** permission have all access permissions, including creating, updating, and deleting dedicated resource pools. Exercise caution when assigning this permission.

Figure 2-10 Assigning permissions for using ModelArts

Assign selected permissions to user_group. ?



- Configure permissions for using other dependent services. For example, to use OBS, search for **OBS** and select **OBS OperateAccess**. ModelArts training jobs

use OBS to forward data. Therefore, the permissions for using OBS are necessary.

For permissions of more cloud services, such as SWR, see [Table 2-2](#).

3. Click **Next** and set the minimum authorization scope. Select **Region-specific projects**, select the region to be authorized, and click **OK**.

In this example, you are allowed to use services in the **CN-Hong Kong** region only.

Figure 2-11 Selecting an authorization scope

Scope

☐ All resources

☒ Region-specific projects

IAM users will be able to use resources in the selected region-specific projects based on assigned permissions.

The selected permissions will be applied to resources in the region-specific projects you select.

Total projects: 12. Select the desired projects.

Project [Region]	Description
☐ af-south-1 [AF-Johannesburg]	--
☒ ap-southeast-1 [CN-Hong Kong]	--

4. A message is displayed, indicating that the authorization is successful. View the authorization information and click **Finish**. It takes 15 to 30 minutes for the authorization to take effect.

Step 3 Create a sub-user. In the navigation pane on the left of the IAM console, choose **Users**. In the right pane, click **Create User** in the upper right corner. On the **Create User** page, add multiple users. Set parameters as prompted and click **Next**.

Step 4 Add the sub-user to the user group. On the **Add User to Group** page, select a user group and click **Create**. The system adds the created users to the user group.

Step 5 [Log in as the IAM user](#) and verify permissions.

Log in to the console as the created user, switch to the authorized region, and verify the permissions.

- Choose **Service List > ModelArts**. In the navigation pane of the ModelArts console, choose your desired type of AI dedicated resource pools and create

one. You should not be able to create a new resource pool if the **ModelArts CommonOperations** permission has taken effect.

- Choose any other service in **Service List**. (Assume that the current policy contains only **ModelArts CommonOperations**.) If a message appears indicating that you have insufficient permissions to access the service, the **ModelArts CommonOperations** permission has taken effect.
- Choose **Service List > ModelArts**. In the navigation pane of the ModelArts console, choose **Asset Management > Datasets** and click **Create**. You should be able to access the corresponding OBS path if the OBS permission has taken effect.
- Verify other optional permissions according to [Table 2-2](#).

----End

Creating a Custom Policy for ModelArts Standard

In addition to the default system permissions of ModelArts, you can create custom policies for refined user permission management, for example, managing OBS operation permissions.

You can create custom policies using either the visual editor or JSON views on IAM. This section describes how to use a JSON view to create a custom policy to grant permissions required to use development environments and the minimum permissions required by ModelArts to access OBS.

For more about other functions and dependent services, see [ModelArts Standard Role/Policy-based Authorization](#).

For details about how to create custom policies and related parameters, see [Creating a Custom Policy](#).

NOTE

A custom policy can contain actions of multiple services that are globally accessible or accessible through region-specific projects.

ModelArts is accessible through region-specific projects, but OBS is globally accessible, so you need to create separate policies for the two services and then apply these policies to the users.

1. Create a custom policy with minimum OBS permissions for ModelArts.
Log in to the IAM console, choose **Permissions > Policies/Roles**, and click **Create Custom Policy**. Set the following parameters:
 - **Policy Name**: Enter a policy name.
 - **Policy View**: Select **JSON**.
 - **Policy Content**: Configure the policy by referring to [Example Custom Policy with OBS Permissions for ModelArts](#).

Figure 2-12 Minimum OBS permissions

★ Policy Name

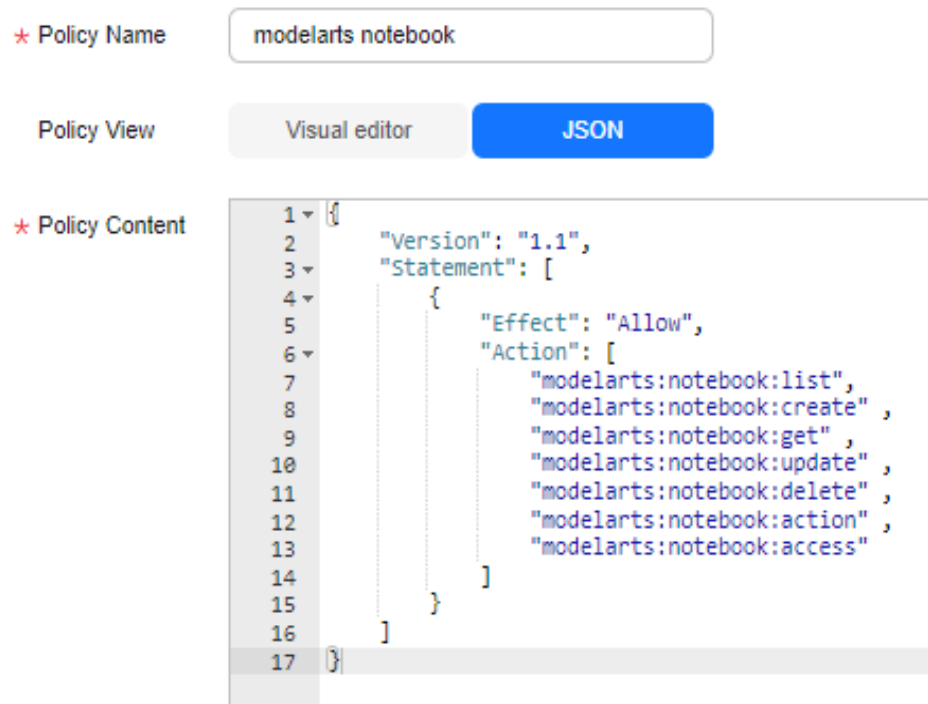
Policy View Visual editor JSON

★ Policy Content

```

1 {
2   "Version": "1.1",
3   "Statement": [
4     {
5       "Action": [
6         "obs:bucket:ListAllMybuckets",
7         "obs:bucket:HeadBucket",
8         "obs:bucket:ListBucket",
9         "obs:bucket:GetBucketLocation",
10        "obs:object:GetObject",
11        "obs:object:GetObjectVersion",
12        "obs:object:PutObject",
13        "obs:object:DeleteObject",
14        "obs:object:DeleteObjectVersion",
15        "obs:object:ListMultipartUploadParts",
16        "obs:object:AbortMultipartUpload",
17        "obs:object:GetObjectAcl",
18        "obs:object:GetObjectVersionAcl",
19        "obs:bucket:PutBucketAcl",
20        "obs:object:PutObjectAcl"
21      ],
22      "Effect": "Allow"
23    }
24  ]
25 }
```

2. Create a custom policy for the permission to use the ModelArts development environment, as shown in [Figure 2-13](#). Set the following parameters:
 - **Policy Name:** Enter a policy name.
 - **Policy View:** Select **JSON**.
 - **Policy Content:** Configure the policy by referring to [Example Custom Policy with Permissions for Using the ModelArts Development Environment](#). For the actions that can be added to ModelArts custom policies, see [ModelArts API Reference > Permissions Policies and Supported Actions](#).

Figure 2-13 Permission to use the development environment

- To authorize the services other than ModelArts and OBS, see [ModelArts Standard Role/Policy-based Authorization](#).
- 3. After creating a user group on the IAM console, grant the custom policy created in **1** to the user group.
- 4. Create a user on the IAM console and add the user to the group created in **3**.
- 5. Log in to the console as the created user, switch to the authorized region, and verify the permissions.
 - Choose **Service List > ModelArts**. In the navigation pane of the ModelArts console, choose **Asset Management > Datasets**. If you cannot create a dataset, the permission granted only for using the development environment has taken effect.
 - Choose **Service List > ModelArts**. In the navigation pane of the ModelArts console, choose **Development & Production > Development Workspace > Notebook** and click **Create Notebook**. If you can access the OBS path specified in **Storage**, the OBS permission has taken effect.

Example Custom Policy with OBS Permissions for ModelArts

In this example, the minimum permissions required by ModelArts to access OBS are assigned, including the permissions for OBS buckets and objects. With these permissions, you can access OBS from ModelArts without restrictions.

```
{
  "Version": "1.1",
  "Statement": [
    {
      "Action": [
        "obs:bucket:ListAllMybuckets",
        "obs:bucket:HeadBucket",
```

```
        "obs:bucket:ListBucket",
        "obs:bucket:GetBucketLocation",
        "obs:object:GetObject",
        "obs:object:GetObjectVersion",
        "obs:object:PutObject",
        "obs:object:DeleteObject",
        "obs:object:DeleteObjectVersion",
        "obs:object:ListMultipartUploadParts",
        "obs:object:AbortMultipartUpload",
        "obs:object:GetObjectAcl",
        "obs:object:GetObjectVersionAcl",
        "obs:bucket:PutBucketAcl",
        "obs:object:PutObjectAcl"
    ],
    "Effect": "Allow"
  }
]
```

Example Custom Policy with Permissions for Using the ModelArts Development Environment

```
{
  "Version": "1.1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "modelarts:notebook:list",
        "modelarts:notebook:create",
        "modelarts:notebook:get",
        "modelarts:notebook:update",
        "modelarts:notebook:delete",
        "modelarts:notebook:action",
        "modelarts:notebook:access"
      ]
    }
  ]
}
```

Related References

Here are some other permission configuration examples for your reference.

- [Viewing All Notebook Instances of an IAM Project](#)
- [Separately Assigning Permissions to Administrators and Developers](#)
- [Prohibiting a User from Using a Public Resource Pool](#)
- [Granting SFS Turbo Folder Access Permissions to a Sub-User](#)

2.2 Creating and Managing a Workspace

ModelArts workspaces enable enterprise customers to split their resources into multiple spaces that are logically isolated and to manage access to different spaces. After workspace is enabled, a default workspace is created. All resources you have created are in this workspace. You can create multiple workspaces. Each acts as a separate ModelArts instance. Workspaces do not interfere with each other, making management easier.

Workspaces support the following access control modes:

- **PUBLIC:** publicly accessible to tenants (including both the tenant account and all its IAM accounts)

- **PRIVATE:** accessible only to the creator and the tenant account
- **INTERNAL:** accessible to the creator, the tenant account, and specified IAM users. When the authorization type is set to **INTERNAL**, specify one or more accessible IAM users.

A default workspace is allocated to each IAM project of each account. The access control of the default workspace is **PUBLIC**.

Scenario

Workspace access control allows the access of only certain users. This function can be used in the following scenarios:

- **Education:** A teacher allocates an **INTERNAL** workspace to each student and allows the workspace to be accessed only by specified students. In this way, students can separately perform experiments on ModelArts.
- **Enterprises:** An administrator creates a workspace for production tasks and allows only O&M personnel to use the workspace, and creates a workspace for routine debugging and allows only developers to use the workspace. In this way, different enterprise roles can use resources only in a specified workspace.

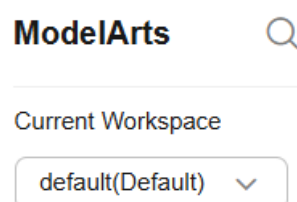
Prerequisites

You have configured the basic permissions for using ModelArts. For details, see [Configuring Agency Authorization for ModelArts with One Click](#).

Choosing a Workspace

1. Log in to the ModelArts console.
2. Switch the workspace in the upper left corner of the console.
 - The system creates a **default** workspace, binds the **default** enterprise project, and sets the authorization type to **PUBLIC**.
 - If the workspace does not meet service needs, you can [Creating a Workspace](#) or [Modifying a Workspace](#).

Figure 2-14 Switching the workspace



Creating a Workspace

1. Log in to the ModelArts console.
2. In the navigation pane on the left, choose **Workspace**.
3. Click **Create workspace**.

Table 2-3 Creating a workspace

Parameter	Description
Workspace name	Enter the name of the workspace. This parameter is mandatory. The value can contain 4 to 64 visible characters, including letters, digits, hyphens (-), and underscores (_).
Description	Enter the description of the workspace, with a maximum of 256 characters.
Enterprise Project	Select an enterprise project. This parameter is mandatory. If no enterprise project is available, click Create Enterprise Project to go to the Enterprise Project Management page, create an enterprise project, and bind it to the workspace. Enterprise projects provide a cloud resource management mode, in which cloud resources and members are centrally managed by project.
Authorization Type	Select the access control mode for the workspace. • PUBLIC: publicly accessible to tenants (including both the tenant account and all its sub-accounts) • PRIVATE: accessible only to the creator and the tenant account • INTERNAL: accessible to the creator, the tenant account, and specified sub-users When Authorization Type is set to INTERNAL, you need to set Authorized User and Authorized To to specify the sub-users who can access the workspace. When Authorized User is set to IAM user, select one or more IAM users in Authorized To. If Authorized User is set to Federated user, enter one or more usernames or IDs of federated users in Authorized To. When Authorized User is set to Agency, select one or more agencies in Authorized To.

4. After setting the workspace parameters, click **Create Now**.

Managing Workspace Quotas

After a workspace is created, you can view and modify the quota information.

1. In the navigation pane of the ModelArts console, choose **Workspace**.
2. On the **Workspace** page, locate the target workspace, and click **Quota Management** in the **Operation** column to go to the workspace details page.
3. In the **Quota Info** area, view the quota value, used quota, and last modification time.

4. Click **Modify Quota** on the right of **Quota Info** to change the quota value. For details about the configurations, see [Table 2-4](#).

Table 2-4 Quota information

Quota	Value Description	Unit
ExeML training duration (predictive analytics)	The value ranges from 1 to 60000 and is unlimited by default.	Minute
ExeML training duration (image classification, object detection, and sound classification)	The value ranges from 1 to 60000 and is unlimited by default.	Minute
GPU-based training job duration (calculated based on a single Pnt1 per node)	The value ranges from 1 to 60000 and is unlimited by default.	Minute
CPU-based training job duration (calculated based on a single vCPU per node)	The value ranges from 1 to 60000 and is unlimited by default.	Minute
Visualization job duration	The value ranges from 1 to 60000 and is unlimited by default.	Minute
CPU-based development environment duration (calculated based on the number of vCPUs)	The value ranges from 1 to 60000 and is unlimited by default.	Minute
GPU-based development environment duration (calculated based on the number of Pnt1 GPUs)	The value ranges from 1 to 60000 and is unlimited by default.	Minute
CPU-based inference service duration (calculated based on the number of nodes)	The value ranges from 1 to 60000 and is unlimited by default.	Minute
GPU-based inference service duration (calculated based on the number of nodes)	The value ranges from 1 to 60000 and is unlimited by default.	Minute
vCPUs for CPU-based training jobs	The value ranges from 1 to 10000 and is unlimited by default.	vCPU

Quota	Value Description	Unit
Cards for GPU-based training jobs	The value ranges from 1 to 1000 and is unlimited by default.	Card
RAM size for training jobs	The value ranges from 1 to 100000 and is unlimited by default.	GB
GPU-based auto labeling duration	The value ranges from 1 to 60000 and is unlimited by default.	Minute

5. After changing the quota values of the workspace, click **Submit**. If data in the **Quota Value** column is refreshed, the quotas have been modified successfully.

Modifying a Workspace

After a workspace is created, you can modify its information.

1. In the navigation pane of the ModelArts console, choose **Workspace**.
2. On the **Workspace** page, locate the target workspace and click **Modify** in the **Operation** column.

You can modify the name, description, enterprise project, and authorization type of the workspace. For details, see [Table 2-4](#).

3. After modifying the parameters, click **Modify Now**.

Deleting a Workspace

You can delete a workspace that is no longer needed. Before deleting a workspace, delete the resource pools and networks created in the workspace.

NOTE

Deleting a workspace will delete all resources in the workspace, including created notebooks, training jobs, and deployed services. The deleted resources cannot be restored.

1. In the navigation pane of the ModelArts console, choose **Workspace**.
2. In the workspace list, click **Delete** in the **Operation** column. In the displayed dialog box, confirm the information about the workspace you want to delete.
If there are resource pools and networks that have not been deleted in the workspace, delete them first.

Confirm the information, enter **DELETE**, and click **OK**. The workspace status changes to **Deleting**. After the resources are cleared, the workspace is deleted from the list.

2.3 Creating an OBS Bucket for ModelArts to Store Data

ModelArts does not provide data storage. Instead, it uses Object Storage Service (OBS) to store data, and backs up and takes snapshots for models, achieving secure, reliable, and cost-effective storage.

All the input data, output data, and cache data during AI development can be stored in OBS buckets for reading. Before using ModelArts, create an OBS bucket and folders for storing data.

Figure 2-15 Interaction between ModelArts and OBS

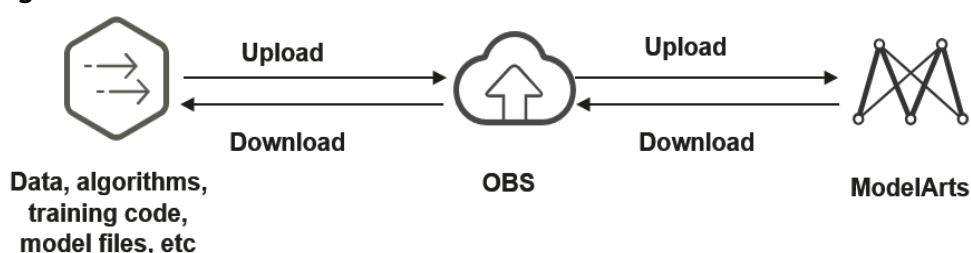


Table 2-5 Relationship between ModelArts and OBS

Function	Subtask	Relationship
Standard workflow	Data labeling	The data labeled on ModelArts is stored in OBS.
	Auto training	After a training job is complete, the generated model is stored in OBS.
	Model deployment	ModelArts deploys models stored in OBS as real-time services.
ModelArts Standard AI development lifecycle	Data management	• Datasets are stored in OBS. • The dataset labeling information is stored in OBS. • Data can be imported from OBS.
	Development environment	Data or code files in a notebook instance can be stored in OBS.
	Model training	Datasets, algorithms, running scripts, training outputs, and training logs used by training jobs can be stored in OBS.

Function	Subtask	Relationship
	Inference deployment	After a training job is complete, the generated model can be stored in OBS. You can import the model from OBS when creating a model.

Procedure for Creating an OBS Bucket

1. Log in to the [OBS console](#) and click **Create Bucket** in the upper right corner of the page to create an OBS bucket.

Figure 2-16 Creating a bucket

Buckets / Create Bucket

<  Create Bucket

General Configuration

Replicate Settings from Existing Bucket

Select Bucket

Optional. The following bucket configurations can be replicated: region, data redundancy

Region

 CN-Hong Kong

Once a bucket is created, its region **cannot** be changed. Resources are region-specific and are stored in the region nearest where your bucket will be accessed. [Learn how to select a region.](#)

Bucket Settings

Bucket Name

c-flowers [View Naming Rules](#)

Bucket domain name: c-flowers.obs.ap-southeast-1.myhuaweicloud.com

NOTE

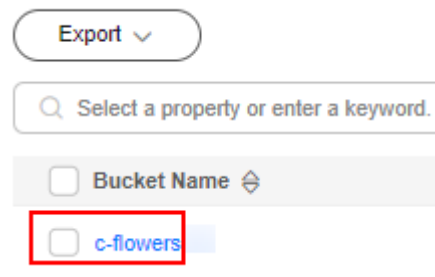
The created OBS bucket and ModelArts are in the same region. For example, if ModelArts is located in the CN-Hong Kong region, select CN-Hong Kong when creating an OBS bucket.

For details about how to obtain the region where the OBS bucket and ModelArts are located, see [Check whether the OBS bucket and ModelArts are in the same region.](#)

Do not enable **Default Encryption**. ModelArts cannot read the data from encrypted OBS buckets.

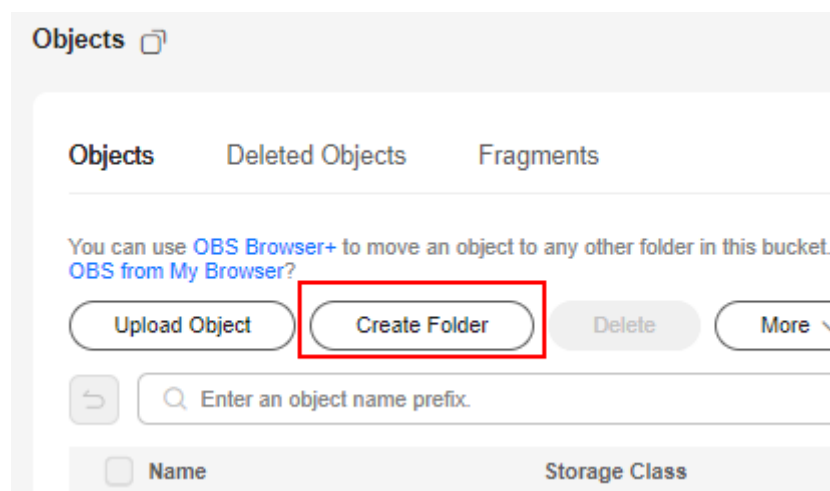
2. In the bucket list, click the bucket name to access its details page.

Figure 2-17 Bucket list



3. In the navigation pane on the left, choose **Objects**. On the **Objects** page, click **Create Folder** to create an OBS folder. For example, create a folder named **flowers** in the created **c-flowers** OBS bucket.

Figure 2-18 Creating a folder



After creating a folder in the OBS bucket, you can upload files. For details about how to upload files, see [Upload Overview](#).

FAQs

- Why can't I find my created OBS bucket after I select an OBS path in ModelArts?
- How do I check whether ModelArts and an OBS bucket are in the same region?
- What should I do if "Error: stat:403" is reported when I perform operations on an OBS bucket?

To resolve the preceding problems or other OBS path exceptions, see [Incorrect OBS Path on ModelArts](#).

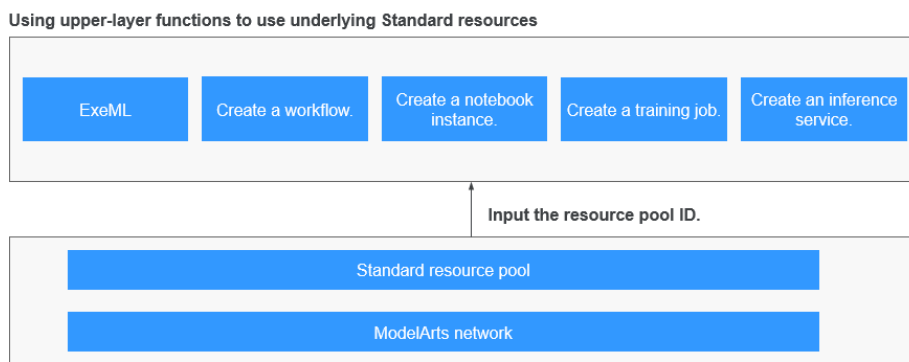
3 ModelArts Standard Resource Management

3.1 About ModelArts Standard Resource Pools

A resource pool centrally manages and allocates compute (CPU and memory), storage (disk space), and network (bandwidth and IP address) resources, ensuring efficient use of resources, flexible allocation, and dynamic management.

A ModelArts Standard resource pool provides compute resources required for AI development (including workflow creation, notebook instance creation, training job creation, and inference service creation) using ModelArts. You can purchase ModelArts Standard resource pools based on service requirements.

Figure 3-1 Using a standard resource pool for AI development



ModelArts Standard Resource Pools

When using ModelArts for AI development, you can use either of the following resource pools:

- **Dedicated resource pool:** It delivers more controllable resources and cannot be shared with other users. Create a dedicated resource pool and select it during AI development.
- **Public resource pool:** It provides large-scale public computing clusters, which are allocated based on job parameter settings. Resources are isolated by job.

You can use ModelArts public resource pools to deliver training jobs, deploy models, or run DevEnviron instances and will be billed on a pay-per-use basis.

Differences between **dedicated resource pools** and **public resource pools**:

- Dedicated resource pools provide dedicated computing clusters and network resources for users. The dedicated resource pools of different users are physically isolated, while public resource pools are only logically isolated. Compared with public resource pools, dedicated resource pools feature better performance in isolation and security.
- When a dedicated resource pool is used for creating jobs and the resources are sufficient, the jobs will not be queued. When a public resource pool is used for creating jobs, there is a high probability that the jobs will be queued.
- A dedicated resource pool is accessible to your network. All running jobs in the pool can access storage and resources in your network. For example, if you select a dedicated resource pool with an accessible network when creating a training job, you can access SFS data after the training job is created.
- Dedicated resource pools allow you to customize the runtime environment of physical nodes, for example, you can upgrade GPU or Ascend drivers. This function is not supported by public resource pools.

Instructions of Dedicated Resource Pools

- If you are using dedicated resource pools for the first time, get started by reading this section.
- Create a dedicated resource pool by referring to [Creating a Standard Dedicated Resource Pool](#).
- View the details about a created dedicated resource pool by referring to [Viewing Details About a Standard Dedicated Resource Pool](#).
- If the specifications of a dedicated resource pool do not meet your service requirements, adjust the specifications by referring to [Resizing a Standard Dedicated Resource Pool](#).
- On the dedicated resource pool list page, you can select an accelerator card driver and perform changes upon submission or smooth upgrade of the driver based on service requirements. Upgrade the GPU/Ascend driver of your dedicated resource pools by referring to [Upgrading the Standard Dedicated Resource Pool Driver](#).
- Rectify the faulty nodes as needed by referring to [Rectifying a Faulty Node in a Standard Dedicated Resource Pool](#).
- Set or change job types supported by a dedicated resource pool by referring to [Modifying the Job Types Supported by a Standard Dedicated Resource Pool](#).
- Administrators can isolate permissions on resources within workspaces for IAM users. You can move resources to the corresponding workspace by referring to [Migrating Standard Dedicated Resource Pools and Networks to Other Workspaces](#).
- Manage resource pools by tag by referring to [Using TMS Tags to Manage Resources by Group](#).
- If a dedicated resource pool is no longer needed, delete it by referring to [Releasing Standard Dedicated Resource Pools and Deleting the Network](#).

3.2 Creating a Standard Dedicated Resource Pool

To meet unique service needs, separate resource pools are needed to guarantee resource availability, security, and optimal performance.

A dedicated resource pool is a set of resources reserved for specific users, services, or projects. It physically or logically isolates these resources from others to meet unique needs.

A ModelArts Standard resource pool provides compute resources required for AI development (including workflow creation, notebook instance creation, training job creation, and inference service creation) using ModelArts. For more information, see [About ModelArts Standard Resource Pools](#).

To get dedicated compute resources, you must create dedicated resource pools within elastic clusters.

This section describes how to create a standard dedicated resource pool.

Billing

When creating a Standard dedicated resource pool, compute resources are billed for running development environments, training jobs, and deployed services. For details, see [Table 3-1](#).

Table 3-1 Billing items

Billing Item		Description	Billing Mode	Billing Formula
Compute resource	Dedicated resource pools	Usage of compute resources. For details, see ModelArts Pricing Details .	Pay-per-use	Specification unit price x Number of compute nodes x Usage duration
			Yearly/Monthly	Specification unit price x Number of compute nodes x Purchase duration

Prerequisites

Ensure that you have enough resource quotas. If the current quota cannot meet your requirements, you can apply for a higher quota. For details, see [Quotas](#).

Log in to the [ModelArts console](#). In the navigation pane on the left, choose **Standard Cluster** under **Resource Management**. You can view the quota (available/total) in the upper right corner of the cluster list.

Step 1: Creating a Network

A ModelArts network connects nodes within a ModelArts resource pool over Huawei Cloud VPC. When creating a network, you only need to choose its name

and CIDR block. To avoid CIDR block conflicts, ModelArts networks offer multiple options for selection.

A VPC provides a logically isolated virtual network for your instances. You can configure and manage the network as required. VPC provides logically isolated, configurable, and manageable virtual networks for cloud servers, cloud containers, and cloud databases. It helps you improve cloud service security and simplify network deployment.

Each user can create a maximum of 15 networks. To create a network, perform the following steps:

1. Log in to the [ModelArts console](#). In the navigation pane on the left, choose **Network** under **Resource Management**. On the displayed page, click **Create Network**.
2. In the **Create Network** dialog box, set parameters.
 - **Network Name**: Enter a custom name or use the default name.
 - **CIDR Block**: Select **Preset** or **Custom**. Recommended CIDR blocks for a custom network: 10.0.0.0/8-24, 172.16.0.0/12-24, and 192.168.0.0/16-24. The subnet mask ranges from 8 to 28. If you need to use an inference service, do not use the following CIDR blocks to avoid conflicts between the VPC CIDR block and the CIDR block reserved for inference: 192.168.0.0/16, 172.16.0.0/16, 10.247.0.0/16, 10.255.0.0/16, 172.16.0.0/16, and 10.247.0.0/16.

Ensure that the network segment does not overlap with the CIDR block of the VPC that you want to access. The network segment cannot be modified once the network is created. Possible conflict CIDR blocks are as follows:

- Your VPC CIDR block
- Container CIDR block (fixed to 172.16.0.0/16)
- Service CIDR block (fixed to 10.247.0.0/16)

Figure 3-2 Creating a network

Create Network

⚠ If you want to use the inference service, ensure that the VPC CIDR block does not conflict with the reserved inference CIDR block:
192.168.0.0/16, 172.16.0.0/16, 10.247.0.0/16, 10.255.0.0/16, 172.16.0.0/16, 10.247.0.0/16.

Network Name: network-246a

Network Segment Type: **Preset** Custom

192. [dropdown] ⓘ

The VPC network segment, container network segment (fixed at 172.16.0.0/16), service network segment (fixed at 10.247.0.0/16), and created VPC network segment can be displayed only in the same workspace.

IPv6 ⓘ ☐ Enable IPv6

Cancel OK

3. Confirm the settings and click **OK**.

You can view the created network on the network management page.

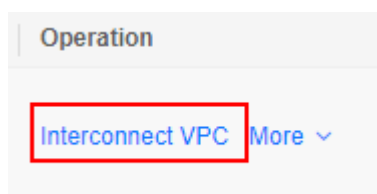
(Optional) Step 2: Interconnecting with a VPC

To interconnect multiple VPCs, you need to establish network links between them or between a VPC and another network environment. This enables resource sharing and secure communication between VPCs.

With VPC interconnection, you can access resources across VPCs, share resources efficiently, improve data transfer speed, and maintain continuous services.

Before interconnecting a VPC, you need to create the VPC and subnet to be interconnected. For details, see [Creating a VPC with a Subnet](#).

1. On the **Network** page, locate the target network click **Interconnect VPC** in the **Operation** column.

Figure 3-3 Interconnecting the VPC

2. In the displayed dialog box, click the button on the right of **Interconnect VPC**, and select an available VPC and subnet from the drop-down lists.

NOTE

The peer network to be interconnected cannot overlap with the current CIDR block.

Figure 3-4 Parameters for interconnecting a VPC with a network



- If no VPC is available, click **Create VPC** on the right to create a VPC.
- If no subnet is available, click **Create Subnet** on the right to create a subnet.
- A VPC can interconnect with at most 10 subnets. To add a subnet, click the plus sign (+).
- To enable a dedicated resource pool to access the public network through a VPC, create a SNAT in the VPC, as the public network address is unknown. After the VPC is interconnected, by default, the public address cannot be forwarded to the SNAT of your VPC. To add a default route, submit a service ticket and contact technical support. Then, when you interconnect with a VPC, ModelArts **0.0.0.0/0** is used as the default route. In this case, you do not need to submit a service ticket. Add the default route for network configuration.

Step 3: Creating a Standard Dedicated Resource Pool

To get dedicated compute resources, you must create dedicated resource pools within elastic clusters.

1. Log in to the [ModelArts console](#). In the navigation pane on the left, choose **Standard Cluster** under **Resource Management**.
2. On the **Standard Cluster** page, click **Buy Standard Cluster**, and configure the parameters as follows on the displayed page.

Table 3-2 AI dedicated cluster parameters

Parameter	Sub-Parameter	Description
Billing Mode	-	Select Yearly/Monthly or Pay-per-use. - Yearly/Monthly is a prepaid billing mode in which your subscription is billed based on the required duration. This mode is more cost-effective when the usage duration is predictable. - Pay-per-use is a postpaid billing mode. You are charged for how long you use each ECS. You can purchase or delete such an ECS at any time.
Cluster Flavor	Pool Name	Enter a name. Only lowercase letters, digits, and hyphens (-) are allowed. The value must start with a lowercase letter and cannot end with a hyphen (-).
	Resource Pool Type	You can select Physical or Logical. Physical Actual hardware resource set, including compute resources such as GPUs, NPUs, and CPUs. These resources ensure ModelArts runtime. Before creating a logical subpool, you need to create a physical resource pool first. Physical resource pools support different job types, such as training and inference. Elastic resources are not supported in physical resource pools, which feature higher isolation, physical isolation, dedicated networks, and network connectivity. Logical A logical resource pool is a resource pool divided from a physical resource pool. You can create multiple logical subpools as required, each of which defines a runtime environment. By creating logical pools based on different computing architectures, resources can be allocated and adjusted dynamically at a fast speed. Elastic resources are supported in logical resource pools, which feature faster creation and scaling. If there is no logical specification, Logical is not available.
	Job Type	Standard resource pools provide compute resources required for AI development jobs using ModelArts. Choose DevEnviron, Training Jobs, and Inference Service as needed.

Parameter	Sub-Parameter	Description
	Advanced Cluster Configuration	• Cluster Specifications: Select Default or Custom. When customizing the specifications, you can set the cluster scale and enable HA for controller nodes. – Configure the cluster scale based on the service scenario. The scale refers to the maximum number of instances that can be managed by a resource pool. – Once HA is enabled for controller nodes, the system creates three control plane nodes for your cluster to ensure reliability. If there are 1,000 or 2,000 nodes in the cluster, HA must be enabled. If HA is disabled, only one control plane node will be created for your cluster. After a resource pool is created, the HA status of controller nodes cannot be changed. • Service Node Provisioning: Select Automatic or Manual. Allocate controller nodes in different AZs for disaster recovery (DR). – Automatic: The system randomly allocates controller nodes to AZs for better DR. If the number of available AZs is less than the number of nodes to be created, the nodes will be created in the AZs with abundant resources to preferentially ensure cluster creation. In this case, AZ-level DR may not be ensured. – Manual: You can also specify an AZ for the controller nodes.
Network	MA Network	Specify the network where the resource pool runs. The network can communicate with other cloud service resource instances on the network. The network needs to be set only for physical resource pools. Select a network from the drop-down list. If no network is available, click Create on the right to create one. For details about how to create a network, see Step 1: Creating a Network. If you need to use an inference service, do not use the following CIDR blocks to avoid conflicts between the VPC CIDR block and the CIDR block reserved for inference: 192.168.0.0/16, 172.16.0.0/16, 10.247.0.0/16, 10.255.0.0/16, 172.16.0.0/16, and 10.247.0.0/16.
	IPv6	Whether to enable IPv6 networking. If enabled, you must enable IPv6 for the network bound to the resource pool. For details about how to enable IPv6, see Step 1: Creating a Network . Once enabled, this function cannot be disabled.

Parameter	Sub-Parameter	Description
Default Specifications	CPU Architecture	CPU architecture of the resource type. Currently, the CPU supports x86 and Arm architectures, as well as heterogeneous scheduling for x86 and Arm64. Set these parameters as required. • x86: Select this if GPU resources are used. It applies to most general-purpose computing scenarios and supports a wide software ecosystem. • Arm64: Select this if NPU resources are used. It applies to specific optimization scenarios, such as mobile applications and embedded systems, features low power consumption.
	Instance Specifications Type	Choose CPU, GPU, or Ascend as needed. Select a CPU architecture and then select instance flavors as required. The flavors vary by region. The actual flavors are displayed on the console. • CPU: general-purpose compute architecture, featuring low computing performance and suitable for general tasks. • GPU: parallel compute architecture, features high computing performance, is suitable for parallel tasks and scenarios such as deep learning training and image processing, and supports multi-PU distributed training. • Ascend: dedicated AI architecture, features extremely high computing performance, is suitable for AI tasks and scenarios such as AI model training and inference acceleration, and supports multi-node distributed deployment.
	Instance Specifications	Select the required specifications from the drop-down list. Due to system loss, the available resources are fewer than specified. Contact your account manager to request resource specifications (such as Ascend) in advance. They will enable the specifications within one to three working days. If there is no account manager, submit a service ticket.
	AZ	Select Automatic or Manual as needed. An AZ is a physical region where resources use independent power supplies and networks. AZs are physically isolated but interconnected over an intranet. • Automatic: AZs are automatically allocated. • Manual: Specify AZs for resource pool nodes. To ensure system disaster recovery, deploy all nodes in the same AZ. You can set the number of instances in an AZ.

Parameter	Sub-Parameter	Description
	Instances	Select the number of instances in a dedicated resource pool. More instances mean higher computing performance. If AZ is set to Manual, you do not need to configure Instances. It is a good practice to create no more than 30 instances at a time. Otherwise, the creation may fail due to traffic limiting. For certain specifications, you can purchase instances by rack. The instances you purchase is the number of racks multiplied by rack(6 node). Purchasing a full rack allows you to isolate tasks physically, preventing communication conflicts and maintaining linear computing performance as task scale increases. All instances in a rack must be created or deleted together.
	Advanced Specifications Configuration	If this parameter is enabled, you can set the OS of the instance.
	Storage Configuration	This parameter can be configured only when Resource Pool Type is set to Physical.
	System Disk	After enabling Storage Configuration, you can view the default system disk type, size, and quantity of each instance. Some specifications do not contain system disks. You can set the type and size of system disks when creating a dedicated resource pool.
	Container Disk	After Storage Configuration is enabled, you can view the type, size, and quantity of container disks of each instance. The container disk type can only be local disk or EVS disk and cannot be changed. Some specifications do not contain container disks. You can set the type and size of container disks when creating a dedicated resource pool. Only EVS disks, including common SSD, high I/O, and ultra-high I/O disks, are supported.

Parameter	Sub-Parameter	Description
	Add Container Data Disk	For some specifications, you can mount additional container disks to each instance in the dedicated resource pool. To do so, click the plus sign (+) before Add Container Disk. The attached disks are EVS disks, which will be charged independently. You can set the type, size, and number of disks to be mounted. The actual values are displayed on the console.
	Disk Space	Container space: The data disk space is divided into two parts by default. One part is used to store the Docker/containerd working directories, container image data, and image metadata. The other is reserved for kubelet and emptyDir volumes. You can set the Specify disk space parameter to set the ratio of the sizes of the two partitions. The available container engine space affects image pulls and container startup and running. If the container disk is a local disk, Specify Disk Space is not supported.
	Container Disk Advanced Configuration - Container Engine Space Size	This parameter specifies the size of the space allocated to the pod container. You can specify the container engine space or set the space to unlimited. The default and minimum values are 50 GiB. You must enter an integer for this parameter. The maximum value depends on the specifications, and can be found in the console prompt. Customizing the container engine space does not increase costs. By specifying this parameter, you can limit the disk size used by a single pod job.

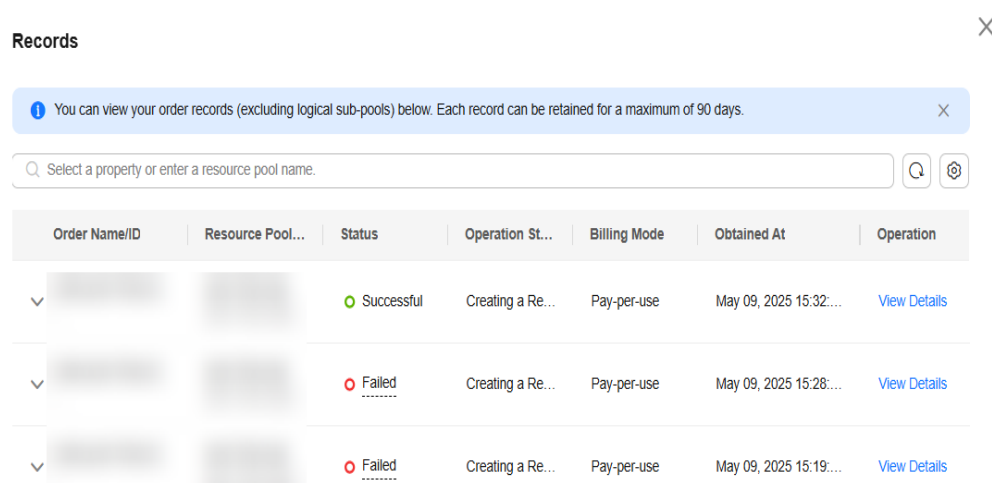
Parameter	Sub-Parameter	Description
	Container Disk Advanced Settings - Write Mode	Some flavors allow you to set the disk write mode, which can be Linear or Stripe. • Linear: A linear logical volume integrates one or more physical volumes. Data is written to the next physical volume when the previous one is used up. • Striped: A striped logical volume stripes data into blocks of the same size and stores them in multiple physical volumes in sequence. This allows data to be concurrently read and written. A storage pool consisting of striped volumes cannot be scaled-out.
	Creating a Flavor	Add multiple specifications as needed. Restrictions: • Each flavor must be unique. • When selecting multiple GPU or NPU specifications, distributed training speed is impacted because different specifications' parameter network planes are not connected. For distributed training, you are advised to choose only one GPU or NPU specification. • You can add up to 10 specifications to a resource pool.

Parameter	Sub-Parameter	Description
Set Plug-in	Select Plug-in	ModelArts provides various plug-ins for you to selectively expand resource pool functions to meet service requirements. The plug-ins installed by default are as follows: ModelArts Node Agent ModelArts Node Agent is a plug-in for monitoring cluster node exceptions, also, a component for connecting to third-party monitoring platforms. It is a daemon that runs on each node to collect node problems from different daemon processes. ModelArts Metric Collector A default built-in plug-in, which runs as a node daemon to collect monitoring metrics of nodes and jobs and report the metrics to AOM. AI Suite (Ascend NPU) Allows containers to use Huawei NPU devices. It is automatically installed when Instance Specifications Type is set to Ascend. Volcano Scheduler Volcano is a batch scheduling platform based on Kubernetes. It provides a series of features required by machine learning, deep learning, bioinformatics, genomics, and other big data applications, as a powerful supplement to Kubernetes capabilities. It is automatically installed when Job Type is set to Training Jobs. Click View Details to view plug-in functions and version updates.
Resource scheduling and allocation	Custom Driver	Disabled by default. Some GPU and Ascend resource pools allow custom driver installation. The driver is automatically installed in the cluster by default. Enable this function if you need to specify the driver version.
	GPU / Ascend Driver	This parameter is displayed if Custom Driver is enabled. You can select a GPU or Ascend driver. The value depends on the driver you choose.

Parameter	Sub-Parameter	Description
Enabling HA redundancy	-	This function can be enabled when snt9C instance specifications are used during resource pool creation. - Enable HA redundancy: Whether to enable HA redundancy for the resource pool. By default, HA redundancy is enabled for supernodes. - Redundant node distribution policy: indicates the distribution policy of redundant nodes. Supernodes support only step-based even distribution. The same number of redundant nodes are reserved in each supernode. - Number of redundant instances: number of HA redundant instances set for this flavor. The redundancy coefficient refers to a quantity of redundant nodes reserved in each supernode when the redundant instance distribution policy is step-based even distribution.
Advanced Options	(Optional) Cluster Description	Enter the cluster description for easy query.
	Tags	Click Add Tag to configure tags for the standard resource pool so that resources can be managed by tag. The tag information can be predefined in Tag Management Service (TMS) or customized. You can also set tag information in the Tags tab of the details page after the standard resource pool is created. Predefined TMS tags are available to all service resources that support tags. Customized tags are available only to the service resources of the user who has created the tags.

Parameter	Sub-Parameter	Description
	CIDR Block	You can select Default or Custom. • Default: The system randomly allocates an available CIDR block to you, which cannot be modified after the resource pool is created. For commercial use, customize your CIDR block. • Custom: You need to customize Kubernetes container and Kubernetes service CIDR blocks. – K8S Container Network: used by the container in a cluster, which determines how many containers there can be in a cluster. The value cannot be changed after the resource pool is created. – Kubernetes Service CIDR Block: CIDR block for services used by containers in the same cluster to access each other. The value determines the maximum number of Services you can create. The value cannot be changed after the resource pool is created. If you need to use an inference service, do not use the following CIDR blocks to avoid conflicts between the VPC CIDR block and the CIDR block reserved for inference: 192.168.0.0/16, 172.16.0.0/16, 10.247.0.0/16, 10.255.0.0/16, 172.16.0.0/16, and 10.247.0.0/16.
Required Duration	-	Select the time length for which you want to use the resource pool. This parameter is mandatory only when the Yearly/Monthly billing mode is selected.
	Auto-renewal	Specifies whether to enable auto-renewal. This parameter is mandatory only when the Yearly/Monthly billing mode is selected. The fees generated by auto-renewal will be deducted from your account balance. For details, see Auto-Renewal. • Monthly subscriptions renew each month. • Yearly subscriptions renew each year.

- Click **Buy Now**. Confirm the information and click **Submit**.
 - After a resource pool is created, its status changes to **Running**.
 - Hover over **Creating** to view the details about the creation process. Click **View Details** to go the operation record page.
 - You can view the task records of the resource pool by clicking **Records** in the upper left corner of the resource pool list.

Figure 3-5 Viewing the resource pool status

Records						
You can view your order records (excluding logical sub-pools) below. Each record can be retained for a maximum of 90 days.						
Select a property or enter a resource pool name.						
Order Name/ID	Resource Pool...	Status	Operation St...	Billing Mode	Obtained At	Operation
▼		Successful	Creating a Re...	Pay-per-use	May 09, 2025 15:32:...	View Details
▼		Failed	Creating a Re...	Pay-per-use	May 09, 2025 15:28:...	View Details
▼		Failed	Creating a Re...	Pay-per-use	May 09, 2025 15:19:...	View Details

FAQ

Why is my dedicated resource pool creation failed, even though I have selected a flavor?

The flavors of dedicated resources change based on real-time availability. Sometimes, you might choose a flavor on the purchase page, but it is sold out before you pay and create the resource pool. This causes the resource pool creation to fail.

You can try a different flavor on the creation page and create the resource pool again.

Why cannot I use all the CPU resources on a node in a resource pool?

Resource pool nodes have systems and plug-ins installed on them. These take up some CPU resources. For example, if a node has 8 vCPUs, but some of them are used by system components, the available resources will be fewer than 8 vCPUs.

You can check the available CPU resources by clicking the **Nodes** tab on the resource pool details page, before you start a task.

Can I use ECSs to create a dedicated resource pool in ModelArts?

No. This operation is not allowed. When creating a resource pool, you can only select available node flavors provided on the console. These node flavors in dedicated resource pools are from ECSs. However, the ECSs purchased under the account cannot be used by the dedicated resource pools for ModelArts.

What are the differences between public resource pools and dedicated resource pool in ModelArts?

- **Dedicated resource pool:** It delivers more controllable resources and cannot be shared with other users. Create a dedicated resource pool and select it during AI development.
- **Public resource pool:** It provides large-scale public computing clusters, which are allocated based on job parameter settings. Resources are isolated by job.

You can use ModelArts public resource pools to deliver training jobs, deploy models, or run DevEnviron instances and will be billed on a pay-per-use basis.

For details, see [What Are the Differences Between a Public Resource Pool and a Dedicated Resource Pool?](#)

What should I do if the resource pool fails to be created?

On the **Standard Cluster** page, click **Records** to view the failure information, and locate the failure cause. For details, see [Failed to Create a Resource Pool](#).

3.3 Managing Standard Dedicated Resource Pools

3.3.1 Viewing Details About a Standard Dedicated Resource Pool

Resource Pool Details Page

1. Log in to the [ModelArts console](#). In the navigation pane on the left, choose **Standard Cluster** under **Resource Management**.
2. You can search for the resource pools by name, ID, resource pool status, node status, resource pool type, and creation time.
3. In the resource pool list, click a resource pool to go to its details page and view its information.
 - If there are multiple ModelArts Standard resource pools, click ▼ in the upper left corner of the details page of one resource pool to switch between resource pools.
 - For a pay-per-use standard resource pool: On the details page, click **More** in the upper right corner to scale in/out, delete a resource pool, change the billing mode to yearly/monthly, and perform other operations. Operations that can be performed vary depending on resource pools.
 - For a yearly/monthly standard resource pool: On the details page, click **More** in the upper right corner to perform operations such as scale-out, unsubscription, renewal, auto-renewal enabling or modification, driver upgrade, and job type setting. Operations that can be performed vary depending on resource pools.
 - In the **Network** area of **Basic Information**, you can click the number of resource pools associated to view associated resource pools. You can view the number of available IP addresses on the network.
 - In the extended information area, you can view the monitoring, jobs, nodes, specifications, events, tags, logical subpools, and plug-ins. For details, see the following section.

Viewing Jobs in a Resource Pool

On the resource pool details page, click **Jobs**. You can view all jobs running in the resource pool. If a job is queuing, you can view its queuing position.

 NOTE

Currently, only training jobs and inference jobs can be viewed.

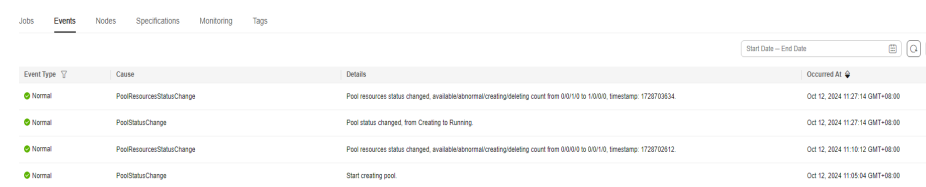
Viewing Resource Pool Events

On the resource pool details page, click **Events**. You can view all events of the resource pool. The cause of an event is **PoolStatusChange** or **PoolResourcesStatusChange**.

In the event list, click  on the right of **Event Type** to filter events.

- When a resource pool starts to be created or becomes abnormal, the resource pool status changes and the change will be recorded as an event.
- When the number of nodes that are available or abnormal or in the process of being created or deleted changes, the resource pool node status changes and the change will be recorded as an event.

Figure 3-6 Events



Event Type	Cause	Details	Occurred At
Normal	PoolResourcesStatusChange	Pool resources status changed, available/abnormal/creating/deleting count from 0/0/1/0 to 1/0/0/0, timestamp: 1726703634	Oct 12, 2024 11:27:14 GMT+08:00
Normal	PoolStatusChange	Pool status changed, from Creating to Running	Oct 12, 2024 11:27:14 GMT+08:00
Normal	PoolResourcesStatusChange	Pool resources status changed, available/abnormal/creating/deleting count from 0/0/0/0 to 0/0/1/0, timestamp: 1726702612	Oct 12, 2024 11:10:12 GMT+08:00
Normal	PoolStatusChange	Start creating pool	Oct 12, 2024 11:05:04 GMT+08:00

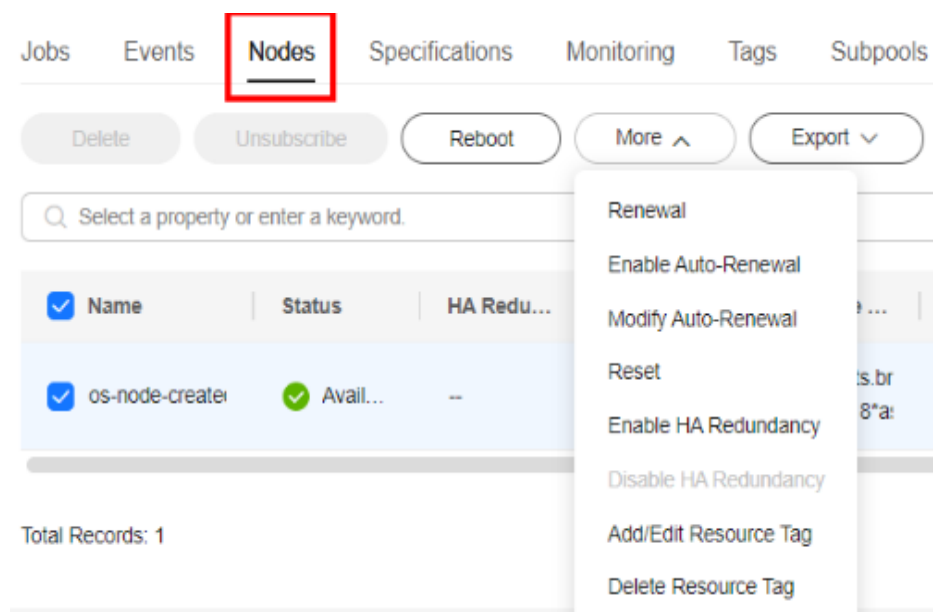
Viewing Resource Pool Nodes

On the resource pool details page, click **Nodes**. You can view all nodes in the resource pool and the resource usage of each node. Hover the cursor over the node to obtain the node name and resource ID. You can use the resource ID to obtain the bills or query the billing information of yearly/monthly resources in the Billing Center.

Some resources are reserved for cluster components. Therefore, **CPUs (Available/Total)** does not indicate the number of physical resources on the node. It only displays the number of resources that can be used by services. CPU cores are metered in milicores, and 1,000 milicores equal 1 physical core.

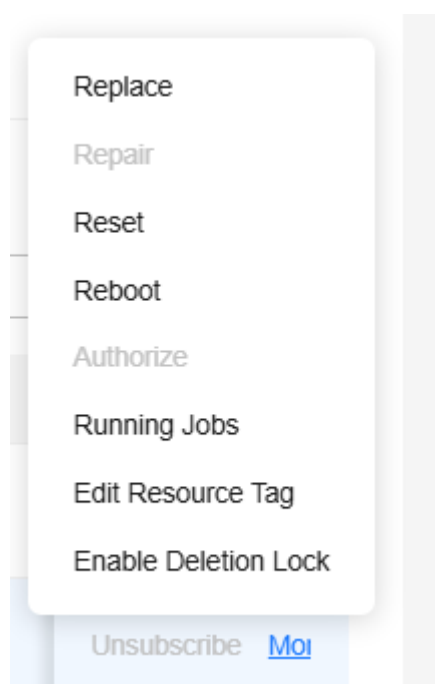
You can delete, unsubscribe from, reset, or restart nodes in batches, or enable or disable HA redundancy for nodes in batches, as shown in the following figure. For details, see [Rectifying a Faulty Node in a Standard Dedicated Resource Pool](#). You can add, edit, and delete resource tags for nodes in batches. You can renew yearly/monthly nodes and enable or modify auto-renewal for them in batches.

Figure 3-7 Performing operations on nodes in batches



To delete, replace, rectify, reset, restart, authorize a node, and enable or disable HA redundancy for a single node, locate the target node in the list and click the button in the **Operation** column. For details, see [Rectifying a Faulty Node in a Standard Dedicated Resource Pool](#). You can view the running jobs of a single node. For details, see [View the Running Jobs of a Resource Pool Node](#). You can view the plug-in resource usage of a single node. For details, see [Querying Plug-in Component](#). You can also edit resource tags.

Figure 3-8 Operations on a single node



In the node search box, you can search for nodes by node name, node status, HA redundancy, batch, driver version, driver status, IP address, and resource tag.

You can export the node information of a standard resource pool to an Excel file. Select the node names, click **Export > Export All Data to XLSX** or **Export >**

Export Part Data to XLSX above the node list, and click  in the browser to view the exported Excel files.

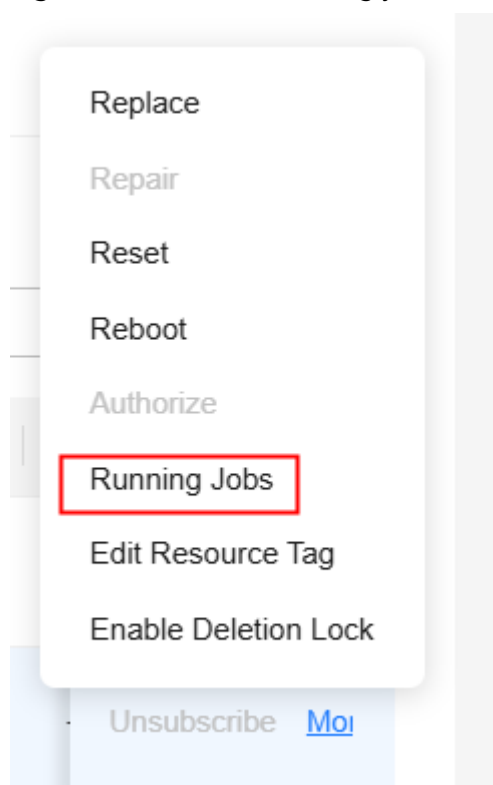
On the node list page, click  to customize the information displayed in the node list.

View the Running Jobs of a Resource Pool Node

In the **Nodes** tab of the resource pool details page, you can view the running jobs of a single node.

To do so, locate the node and choose **More > Running Jobs** in the **Operation** column. In the displayed dialog box, you can view the running jobs of the node by searching for the keyword.

Figure 3-9 View the running jobs of a resource pool node



Querying Plug-in Component

On the resource pool details page, choose **Node Management** from the navigation pane to view the plug-in usage of the current node.

To view the instance usage of the plug-in, locate the target node and choose **More > Query Plug-in Component** in the **Operation** column.

Figure 3-10 Component instances

Component List

Component Name	Namespace	Deployment Mode	Replicas	CPU Quota	Memory Quota
maos-node-agent	default	DaemonSet	--	Applied 100m Maximum 500m	Applied 200Mi Maximum 1024Mi
volcano-admission-6...	kube-system	ReplicaSet	--	Applied 200m Maximum 500m	Applied 500Mi Maximum 500Mi
volcano-controller-5...	kube-system	ReplicaSet	--	Applied 500m Maximum 2000m	Applied 500Mi Maximum 2000Mi

Total Records: 3

10 < 1 >

Viewing Resource Pool Specifications

On the resource pool details page, click **Specifications**. You can view the resource specifications used by the resource pool and the number of resources corresponding to the specifications, and adjust the container engine space.

Figure 3-11 Viewing resource pool specifications

Jobs Events Nodes Specifications Monitoring Tags Subpools

Instance Specification...	Metering ID	Container E...	CPU Cores	CPU Archite...	Memory	AI Accelerator	Quantity	Disk Capacity	Step	Operation
CPU: 8 vCPUs 32GB (...)	maos-vm-cpu-8ud-2596...	50 GiB	8	x86	32GiB	--	1	500GiB	--	Resize Container Engine Space

Viewing Resource Pool Monitoring Information

On the resource pool details page, click **Monitoring**. The resource usage including used CPUs, memory usage, and available disk capacity of the resource pool is displayed. If AI accelerators are used in the resource pool, the GPU and NPU monitoring information is also displayed.

You should set the statistical period to less than seven days. If the period is too long, the query may be delayed due to a large number of AOM metrics.

Figure 3-12 Viewing monitoring metrics

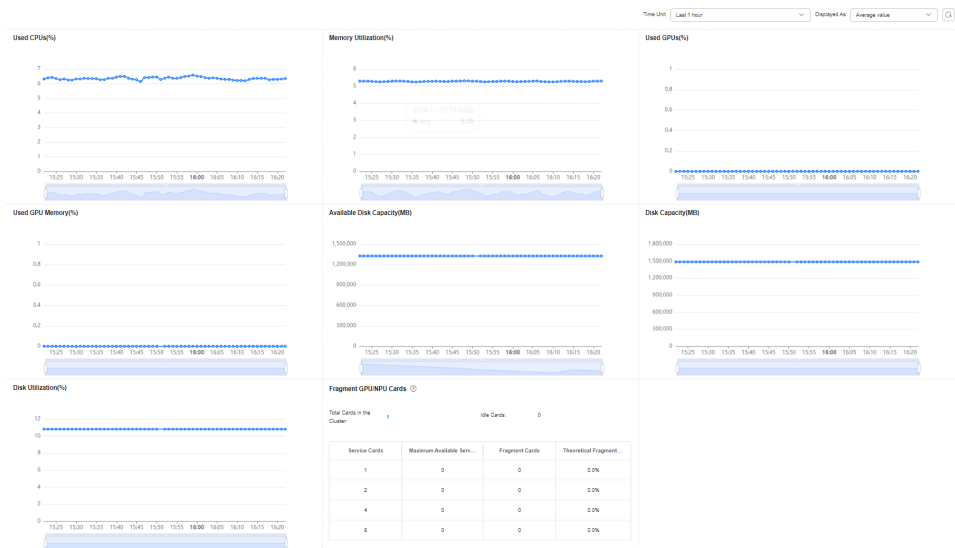


Table 3-3 Monitoring metrics

Parameter	Description	Unit	Value Range
Used CPUs	CPU usage of a measured object	%	0%–100%
Memory Utilization	Percentage of the used physical memory to the total physical memory	%	0%–100%
Used GPUs	Percentage of the used GPUs to the total GPUs	%	0%–100%
Used GPU memory	Percentage of the used GPU memory to the total GPU memory	%	0%–100%
Used NPUs	Percentage of the used NPU memory to the total NPU memory	%	0%–100%
Used NPU Memory	Percentage of the used GPU memory to the total GPU memory	%	0%–100%
Available Disk Capacity	Available disk capacity of a measured object	MB	≥0
Disk Capacity	Total disk capacity of a monitored object	MB	≥0
Disk Utilization	Disk usage of the monitored object	%	0%–100%

Parameter	Description	Unit	Value Range
Fragment GPU/NPU Cards	Fragments are generated during resource scheduling. As a result, some cards are idle but cannot be used by multi-card tasks. For tasks with different numbers of cards, fragments vary according to the distribution of occupied cards and vary with time. The table lists only the status at the current time.	/	/

Managing Resource Pool Tags

You can add tags to a resource pool for quick search.

On the resource pool details page, click **Tags**. You can view, search for, add, modify, and delete tags of a resource pool.

NOTE

You can add up to 20 tags.

Managing Logical Subpools

You can split logical subpools from a ModelArts Standard dedicated resource pool and set resource quotas for each subpool, including reserved quotas and quota rollout. After a logical subpool is split, tasks in the training, inference, and development environments must be delivered to the logical subpool and restricted by the quota. Only tasks within the quota can be delivered and run.

On the resource pool details page, switch to the **Logical Subpool** tab. You can create, resize, and delete a logical subpool, as well as setting the job type for the subpool. For details, see [Managing Logical Subpools of a Standard Dedicated Resource Pool](#).

Binding to a Logical Subpool

1. After [node binding is enabled](#) for the standard dedicated resource pool, nodes in a resource pool can be bound to a specified logical subpool.
2. After a logical subpool is bound, the jobs on the associated nodes may be preempted. To add a node to the target logical subpool, you can move the redundant nodes from the target subpool to the available nodes. For details, see [Migrating a Node](#).
3. Log in to the [ModelArts console](#). In the navigation pane on the left, choose **Standard Cluster** under **Resource Management**.
4. Click the resource pool with node binding enabled. The resource pool details page is displayed.

5. Switch to the **Nodes** tab, select the target nodes, and click **Bind Logical Subpool** above the list.
6. In the displayed dialog box, select the target logical subpool and click **OK**.
For logical subpools with node binding enabled, check whether node draining needs to be enabled during scaling. If node draining is enabled, the drainage will be performed on the nodes to be scaled only this time. You can check the drainage information in the **Nodes** tab. If the draining fails, hover the cursor over the node status to view the failure cause.
Node draining is to clear node resources to release or remove nodes that are no longer needed.

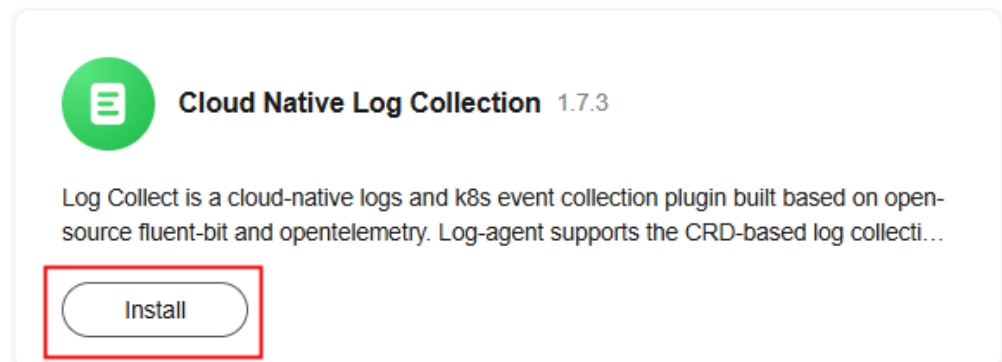
 NOTE

During the draining, nodes cannot be scheduled and job loads on the nodes will be evicted, which may cause job failures.

Viewing Resource Pool Plug-ins

1. ModelArts provides various plug-ins for you to selectively expand resource pool functions to meet service requirements.
2. On the resource pool details page, click the **Plug-ins** tab. You can view the information about all plug-ins of the resource pool.
3. Installing a plug-in: Locate the target plug-in, click **Install**, configure related information, and click **OK**. For details, see [Managing Standard Dedicated Resource Pool Plug-ins](#).

Figure 3-13 Installing a plug-in



1. View plug-in details: Click a plug-in name to view its details, including the introduction and component list.

Figure 3-14 Plug-in details

Plug-in Details



ModelArts Node Agent

7.0.0-2...

Running

Introduction

Component List

Function

The ModelArts Node Agent is a plugin that monitors abnormal events on cluster nodes and a component that interfaces with third-party monitoring platforms. It runs as a daemon on each node, collecting node issues from various daemons.

Created

May 07, 2025 11:05:39 GMT+08:00

Version

7.0.0-20250414160729

Change History

Q

Select a property or enter a keyword.

Q

⚙

Version	New Feature
7.0.0-20250414160729	Kubernetes v1.31 is supported
7.0.0-20250409171907	Kubernetes v1.31 is supported
7.0.0-20250407150406	Kubernetes v1.31 is supported
7.0.0-20250331170107	Kubernetes v1.31 is supported
7.0.0-20250331145906	Kubernetes v1.31 is supported
7.0.0-20250327171246	Kubernetes v1.31 is supported
7.0.0-20250321143651	Kubernetes v1.31 is supported
7.0.0-20250314141623	Kubernetes v1.31 is supported
7.0.0-20250313194413	Kubernetes v1.31 is supported
7.0.0-20250313101711	Kubernetes v1.31 is supported

Checking the Disk Specifications of a Resource Pool

1. On the resource pool details page, choose **More > Scaling** in the upper right corner. On the displayed page, you can view the disk type, size, quantity, write mode, container engine space, and mount path of the system disk, container disk, and data disk.

3.3.2 Resizing a Standard Dedicated Resource Pool

The demand for resources in a dedicated resource pool may change due to the changes of AI development services. In this case, you can resize your dedicated resource pools in ModelArts Standard.

You can add instances to or delete instances from a resource pool for resizing. The instances to be added or deleted must share the same specifications with that of existing instances. This helps you adjust the resource pool scale. For better resource usage, you can delete abnormal or idle instances from a resource pool. For details, see [Viewing Resource Pool Nodes](#).

Issue 01 (2025-08-29)

Copyright © Huawei Cloud Computing Technologies Co., Ltd.

51

WARNING

- When you reduce the number of instances, nodes with deletion lock enabled may be deleted, interrupting running services. This operation cannot be rolled back. Therefore, you are advised not to delete such nodes. If scale-in is still required, go to the resource pool details page to delete the node by referring to [Viewing Resource Pool Nodes](#).
- The scale-in operation may affect running tasks and cannot be rolled back. If the node has a local disk, the data on the local disk will be cleared and cannot be restored. Exercise caution when performing this operation. If the development environment is enabled for the resource pool, the created notebook that uses EVS storage may be unavailable after scale-in.
- Before scaling in a resource pool, ensure that there are no services running in the pool. Alternatively, go to the resource pool details page by referring to [Viewing Resource Pool Nodes](#), delete the nodes where no services are running to scale in the pool.

Billing

When you increase the number of instances, compute resources will be billed. For details, see [ModelArts Pricing Details](#).

You can specify the billing mode of new nodes in a resource pool during scaling. For example, you can create pay-per-use nodes in a yearly/monthly resource pool. If you do not specify this parameter, the billing mode of the created nodes is the same as that of the resource pool. For details, see [Table 3-4](#).

Table 3-4 Billing items

Billing items		Description	Billing Mode	Billing Formula
Compute resource	Dedicated resource pools	Usage of compute resources. For details, see ModelArts Pricing Details .	Pay-per-use	Specification unit price x Number of compute nodes x Usage duration
			Yearly/Monthly	Specification unit price x Number of compute nodes x Purchase duration

Prerequisites

You have [created a standard dedicated resource pool](#).

Constraints

- Only dedicated resource pools in the **Running** status can be resized. A resource pool is suspended when it is being migrated from the old version to

the new version. You cannot adjust the capacity of such a resource pool or unsubscribe from it.

- If there is only one instance node in the resource pool, scaling in cannot be performed. Therefore, keep at least one node.
- The scale-in operation may affect running tasks and cannot be rolled back. If the node has a local disk, the data on the local disk will be cleared and cannot be restored. Exercise caution when performing this operation. If the development environment is enabled for the resource pool, the created notebook that uses EVS storage may be unavailable after scale-in.

Changing the Number of Instances

1. Log in to the [ModelArts console](#). In the navigation pane on the left, choose **Standard Cluster** under **Resource Management**.
2. For a yearly/monthly resource pool, only **Scale Out** is displayed. To scale in the resource pool, go to the details page and unsubscribe from the resources. For a yearly/monthly resource pool, only **Scale Out** is displayed. To scale in the resource pool, go to the details page and unsubscribe from the resources.
3. On the **Scale In/Out Dedicated Resource Pool** page, configure resources by referring to [Table 3-5](#).

Table 3-5 Parameters for resource configurations

Parameter	Description
Instance Specifications Type	Instance specifications type of the target Standard resource pool, which cannot be modified.
Specifications	Specifications of the target Standard resource pool, which cannot be modified.
Instances in Total	Number of instances in the target Standard resource pool, which cannot be modified.
AZ Allocation	AZ allocation of nodes after scaling. You can select Automatic or Manual. • If you select Automatic, nodes are randomly allocated to AZs after the scaling. • If you select Manual, you can allocate nodes to specified AZs. By default, the value of Target Instances indicates how many instances there will be in the AZ. For example: – If there are three instances, and Target Instances is set to 5, the instances in the AZ will be scaled up to 5. – If there are three instances, and Target Instances is set to 2, the instances in the AZ will be scaled in to 2.

Parameter	Description
Container Engine Space Limit	When you scale out a resource pool, and the value of Target Instances is greater than that of Instances in Total, you can set the container engine space size of the new node to a specified value or unlimited. This operation will cause inconsistencies in dockerBaseSize of nodes within the resource pool. As a result, some tasks may run differently on different nodes. The container engine space size cannot be changed for existing nodes.
Container Engine Space Size	You can set a specified size by setting Container Engine Space Limit to Manual .
Target Instances	You can set this parameter for scaling based on service requirements. • Scale out: The value of Target Instances is greater than that of Instances in Total. • Scale in: The value of Target Instances is smaller than that of Instances in Total. If AZ Allocation is set to Manual, you do not need to set this parameter. Target Instances indicates how many instances can there be in the target AZ. When you purchase a resource pool, the instances for certain specifications can be purchased by rack. When you resize the resource pool, the instances are also added or deleted by rack. You can choose to purchase nodes by rack when creating a resource pool, which cannot be modified when resizing a resource pool. You can configure the rack quantity to change the number of target instances.
OS	Select an OS version from the OS drop-down list.

Figure 3-15 Configuring resources on a single node

Resource Configurations

Instance Specifications Type	Ascend 8*ascend-snt9b 100GiB CPU: 192 vCP...	Specifications	192vCPUs 1536GiB 1000GiB
AZ Allocation	☒ Automatic ☐ Manual	Instances in Total	6
Container Engine Space Limit	☒ ☐ Unlimited	Container Engine Space Size	50 GiB
Target Instances	2	Operating system	Preset image EulerOS 2.10 (Current version) ▼

4. Configure the node billing mode. When adding instances, you can enable **Node Billing Mode** to set the billing mode and required duration for the new nodes, and enable the auto-renewal function. For example, you can create

pay-per-use nodes in a yearly/monthly resource pool. If the billing mode is not specified, it will be the mode for charging resource pools.

5. Click **Submit** and then **OK**.

On the **Standard Cluster** page, check whether the number of nodes in the resource pool is the value of **Target Instances**.

Related Operations

- **Viewing Resource Pool Nodes:** If there are abnormal or idle nodes that need to be removed from a resource pool, access the resource pool details page, and delete the target nodes (in batches).
- **Upgrading the Standard Dedicated Resource Pool Driver:** If there are GPU/Ascend resources in the dedicated resource pool, you can upgrade the GPU/Ascend driver based on service requirements.
- **Rectifying a Faulty Node in a Standard Dedicated Resource Pool:** Rectify the faulty nodes in a standard dedicated resource pool. Currently, you can reset and restart a node, as well as configuring an HA redundant node.
- **Modifying the Job Types Supported by a Standard Dedicated Resource Pool:** You can change job types supported by a dedicated resource pool after it is created to avoid resource waste.

3.3.3 Upgrading the Standard Dedicated Resource Pool Driver

Description

If there are GPU/Ascend resources in a dedicated resource pool node, and the node performance cannot meet your requirements, you can upgrade the driver to resolve known issues, improve performance, or support new functions, ensuring resource pool performance and compatibility.

ModelArts allows you to upgrade the GPU/Ascend driver of a dedicated resource pool on the ModelArts console as required.

Secure Upgrade and Forcible Upgrade

There are two driver upgrade modes: secure upgrade and forcible upgrade. The following table describes the comparisons.

Table 3-6 Secure upgrade and forcible upgrade

Item	Secure Upgrade	Forcible Upgrade
Introduction	Upgrade the driver when the node is idle, which does not affect running tasks. The smooth upgrade reduces the impact on services. After the upgrade starts, the nodes will be isolated (new jobs cannot be delivered). Only after the existing jobs on the node are complete will the upgrade be performed. This may take a rather long time as jobs must be completed first.	Running tasks on the node will be ignored and the driver will be directly upgraded. This upgrade mode is fast as you do not need to wait until the node is idle.
Scenario	Non-urgent and gradual upgrade	Urgent and fast upgrade
Notes	The upgrade period is rather long as nodes must be idle first. Before the upgrade, plan the node idle time to reduce the impact on services.	Running tasks may be interrupted or fail. Exercise caution when using this mode.

Constraints

- The target dedicated resource pool must be running, and the resource pool contains GPU or Ascend resources.
- For logical resource pools and logical subpools, you need to [enable node binding](#) for driver upgrade.
- When you upgrade the driver of a standard dedicated resource pool, it will not affect nodes bound to a logical subpool that has upgraded its driver. To upgrade the drivers of these nodes, update the driver of the logical subpool. To upgrade the entire physical pool, [disable node binding for the logical pool](#).
- To upgrade a resource pool, the nodes in it need to be restarted. Perform the upgrade during off-peak hours so that running tasks will not be affected. Check the resource pool node status on the resource pool details page by referring to [Viewing Resource Pool Nodes](#).

Upgrading the GPU/Ascend Driver in a Dedicated Resource Pool

1. Log in to the [ModelArts console](#). In the navigation pane on the left, choose **Standard Cluster** under **Resource Management**.
2. Locate the target resource pool in the list and choose ***** > Upgrade Driver** in the **Operation** column.
3. In the displayed dialog box, you can view the driver type, number of instances, current version, target version, upgrade mode, upgrade scope, and

rolling switch of the dedicated resource pool. Set the parameters by referring to [Table 3-7](#).

Table 3-7 Parameters for upgrading a driver

Parameter	Description
Target Version	Choose the target version from the drop-down list. The driver of the added nodes may not be that of the existing nodes. Select the current driver version for Target Version. After the upgrade, all nodes will be upgraded to the same version
Upgrade Mode	Select Secure upgrade or Forcible upgrade. For details about the differences, see Secure Upgrade and Forcible Upgrade. • Secure upgrade: Perform the upgrade when no job is running on the node. The upgrade may take a long time. • Forcible upgrade: Ignore the running jobs and perform the upgrade directly. This may cause the running jobs to fail.
Rolling	Once enabled, you can upgrade the driver in rolling mode. Rolling upgrade is a gradual instance replacement method that applies to scenarios where service continuity is required. Instances are upgraded in batches to ensure that some instances are running properly during the upgrade, reducing the downtime. During the rolling upgrade, the nodes with abnormal drivers do not affect the upgrade and will also be upgraded.
Rolling Mode	Currently, By node percentage and By instance quantity are supported. • By node percentage: The number of instances to be upgraded is the percentage multiplied by the total number of instances in the resource pool. • By instance quantity: The number of instances to be upgraded is the value of this parameter. For different upgrade modes, the policies for upgrading nodes are different. • If Secure upgrade is selected, the instances without services are upgraded. To check whether a node has any service, go to the resource pool details page. In the Nodes tab, check whether all GPUs and Ascend chips are available. If yes, the node has no services. • If Forcible upgrade is selected, random instances are upgraded.

Parameter	Description
Node Percentage	Set this parameter if Rolling Mode is set to By node percentage . The number of instances to be upgraded in each batch = The value of By node percentage x Total number of instances in the resource pool.
Instances	Set this parameter if Rolling Mode is set to By instance quantity .

Figure 3-16 Upgrading a driver

Upgrade Driver

×

Driver Type

Ascend

Instances

1

Current Version

Target Version

Upgrade Mode

Secure upgrade

Forcible upgrade

?

Upgrade Scope

Rolling

☒

Rolling Mode

By node percentage

By instance quantity

Node Percentage

−

25

+

%

Cancel

OK

- Click **OK** to start the driver upgrade.

In the resource pool list, locate the target resource pool, and choose **More > Upgrade Driver**. On the displayed page, check whether the current version is the target version. If yes, the driver is upgraded.

FAQ

How do I locate a faulty node in a standard resource pool?

In a standard resource pool, ModelArts will add a taint to a faulty Kubernetes node so that jobs will not be scheduled to the tainted node. For details, see [Faulty Nodes in a Standard Resource Pool](#).

3.3.4 Rectifying a Faulty Node in a Standard Dedicated Resource Pool

This section describes how to rectify the faulty nodes. Currently, you can reset and replace the node, as well as configuring an HA redundant node. During fault locating and performance diagnosis, certain O&M operations need to be authorized by users.

HA Redundant Node

- Function

An HA redundant node is the backup node in a dedicated resource pool. It can automatically replace a faulty common node to improve the overall availability and fault tolerance capability of the resource pool, preventing service loss caused by the failure of a single node and ensuring service continuity and stability. You can set the number of HA instances based on your requirements.

 WARNING

HA redundant nodes cannot be used for service running, which will affect the number of actual available instances in the resource pool. When a resource pool delivers a task, select the number of actual available instances with caution. If the HA redundant instances are included, the task will fail to be delivered.

- Mechanism

- HA redundant nodes are isolated and cannot be scheduled by default. Workloads cannot be scheduled to the nodes.
- HA redundant nodes function as standby nodes and work with node fault detection to ensure that the faulty nodes are automatically replaced by normal nodes in a resource pool. The replacement normally takes only a few minutes. After the replacement, the original HA redundant nodes will be de-isolated and become normal nodes, and the faulty nodes will be labeled as HA redundant nodes. Rectify the fault for future automatic switchover. After the faulty nodes are rectified, they become the new HA redundant nodes.

HA redundant nodes can free you from paying attention to node status and reduce O&M costs. However, you need to purchase backup nodes as HA redundant nodes. The resource costs are higher.

NOTE

If a logical sub-pool is created in a Standard physical pool and the sub-pool is transparent, HA redundancy is no longer available in the Standard physical pool and is only available in the sub-pool. The transparency is a white-list function. Submit a service ticket to use it.

Setting HA nodes: You can set multiple HA redundant nodes in batches for a resource pool or set a single node as an HA redundant node.

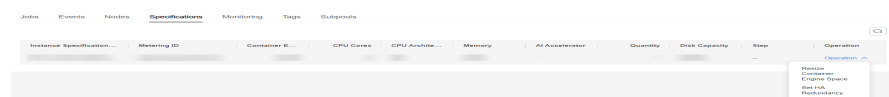
5% of the total nodes in a resource pool should be HA redundant nodes. For example, there are 20 nodes in a resource pool, then there should be one HA redundant node.

If a service node is set as an HA redundant node, the jobs on the node still occupy the actual available resources of the resource pool. However, HA redundant nodes are not counted in the actual available resources, and the result may affect the node scheduling of the actual available resources. Note that the actual available resources of a resource pool are the total available resources of the resource pool minus the resources occupied by HA redundant nodes. When setting HA redundant nodes, use service-free nodes.

To check whether a node has any service, go to the resource pool details page. In the **Nodes** tab, check whether all GPUs and Ascend chips are available. If yes, the node has no services.

- Setting multiple HA redundant nodes in batches for a resource pool
 - Method 1: Set when purchasing the license (supported only by Snt9C).
Parameters:
 - Enable HA redundancy: Whether to enable HA redundancy for the resource pool. By default, HA redundancy is enabled for supernodes.
 - Redundant node distribution policy: indicates the distribution policy of redundant nodes. Supernodes support only step-based even distribution. The same number of redundant nodes are reserved in each supernode.
 - Number of redundant instances: number of HA redundant instances set for this flavor. The redundancy coefficient refers to a quantity of redundant nodes reserved in each supernode when the redundant instance distribution policy is step-based even distribution.
 - Method 2: Set on the resource pool details page.

Figure 3-17 Setting in the **Specifications** tab



- Method 3: Set on the **Scaling** page.
- Configure a single node as HA redundant node.
 - Enabling HA redundancy

Select a node without any service as the HA redundant node. On the resource pool details page, click the **Nodes** tab, locate the target node, and click **More > Enable HA Redundancy** in the **Operation** column.

If you need to enable HA redundancy for nodes in batches, select the target nodes, and click **Enable HA Redundancy** above the list.

Figure 3-18 Enabling HA redundancy

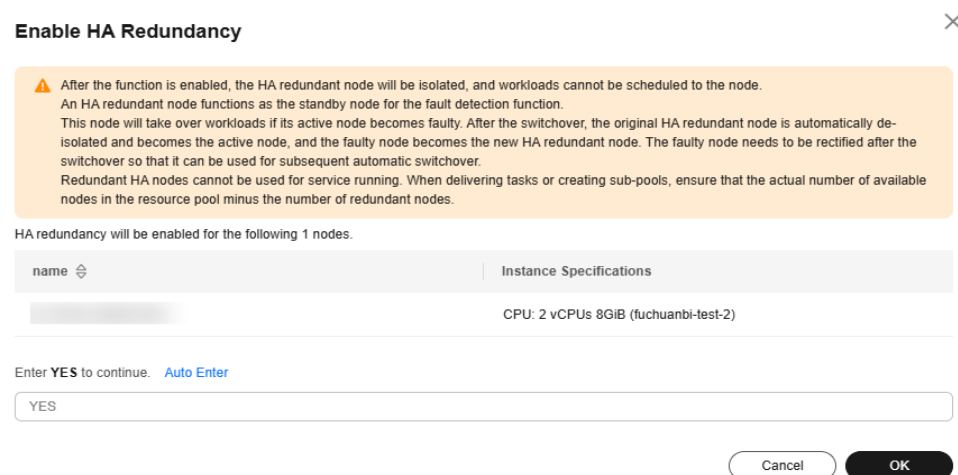
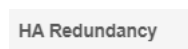


Figure 3-19 HA redundant node



– Disabling HA redundancy

On the resource pool details page, click the **Nodes** tab. Locate the target node and click **More > Disable HA Redundancy** in the **Operation** column.

Once disabled, the nodes will be de-isolated and no longer used as standby nodes. Workloads can be scheduled to the nodes properly.

If you need to disable HA redundancy for nodes in batches, select the target nodes, and click **Disable HA Redundancy** above the list.

Figure 3-20 Disabling HA redundancy**Disable HA Redundancy**Enter **YES** to continue. [Auto Enter](#)

YES

Cancel

OK

Figure 3-21 Non-HA redundancy

HA Redundancy

--

Resetting a Node

Perform this operation to upgrade the OS of the node. If an error occurs during node configuration update, reset the node to rectify the fault.

To reset a single node, access the **Nodes** tab, locate the target node in the list and click **Reset** in the **Operation** column. You can also select multiple nodes, and click **Reset** to reset multiple nodes.

NPU nodes whose instance specifications are snt9b23 cannot be reset.

Configure the parameters described in the table below.

Table 3-8 Parameters for resetting a node

Parameter	Description
Operating system	Select an OS from the drop-down list.
Configuration Mode	Select a configuration mode for resetting the node. • By node percentage: the maximum percentage of nodes that can be reset at a time • By instance quantity: the maximum number of nodes that can be reset at a time

Figure 3-22 Resetting a node

Reset Node

⚠

Resetting a node will affect the running of related services. Exercise caution when performing this operation.

You are about to reset the following 1 nodes.

Name	Status	Operating system
	Available	Huawei Cloud EulerOS 2.0

Operating system

Preset image

Huawei Cloud EulerOS 2.0 Recommended

Configuration Mode

By node percentageBy instance quantity

Node Percentage

25%

Enter YES to continue. Auto Enter

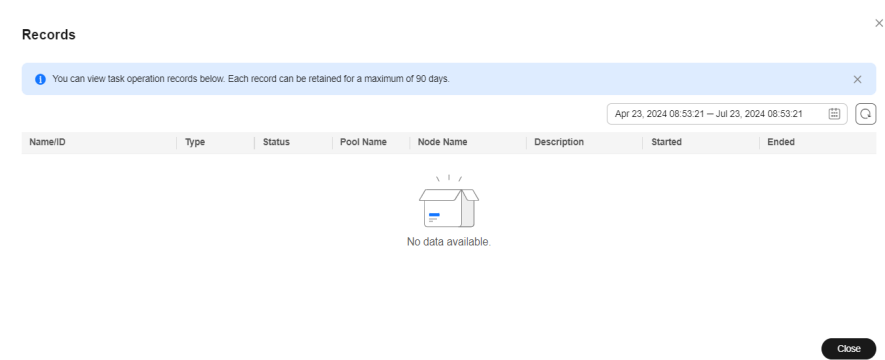
YES

Check the node reset records on the **Records** page. If the node is being reset, its status is **Resetting**. After the reset is complete, the node status changes to **Available**. Resetting a node will not be charged.

NOTE

- Resetting a node will affect running services.
- Only nodes in the **Available** or **Unavailable** state can be reset.
- A single node can be in only one reset task at a time. Multiple reset tasks cannot be delivered to the same node at a time.
- If there are any nodes in the **Replacing** state in the operation records, nodes in the resource pool cannot be reset.
- When the driver of a resource pool is being upgraded, nodes in this resource pool cannot be reset.
- After the node is reset, it will be unavailable for a short time. The service is being started and the health status is being checked. Wait patiently.
- For GPU and NPU specifications, after the node is reset, the driver of the node may be upgraded. Wait patiently.

Figure 3-23 Operation records

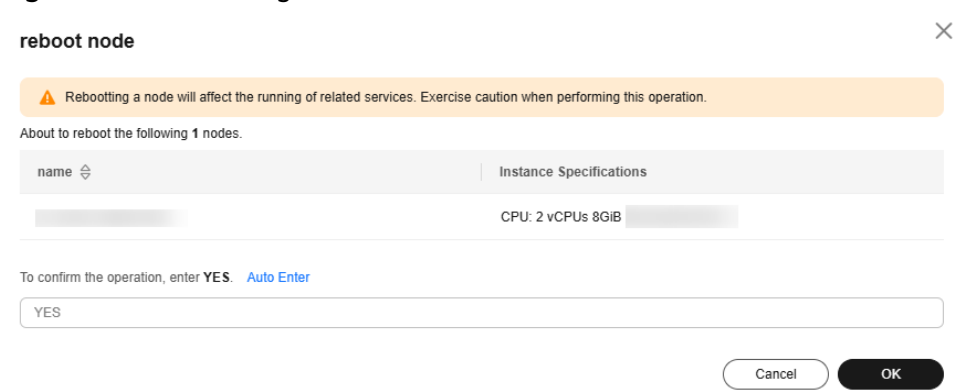


Restarting a Node

On the resource pool details page, click the **Nodes** tab, you can restart a node. To restart a single node, locate the target node, and click **More > Reboot** in the **Operation** column. You can also select multiple nodes and click **Reboot** above the list to restart multiple nodes.

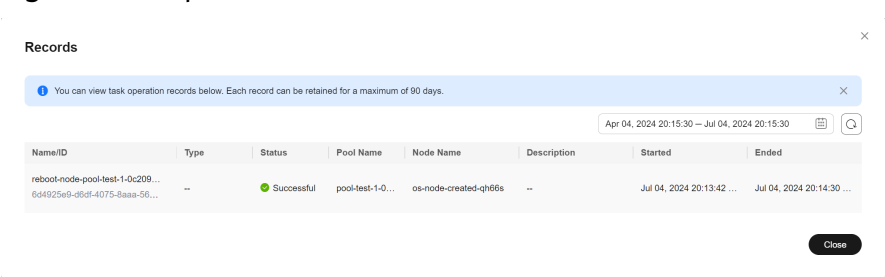
When delivering a node restart task, select the corresponding nodes. Restarting a node will affect running services. Exercise caution.

Figure 3-24 Restarting a node



Check the node operation records in the **Records** dialog box. If the node is being restarted, its status is **Rebooting**. After it is restarted, the status changes to **Available**. Restarting a node will not be charged.

Figure 3-25 Operation records



 NOTE

- Restarting a node will affect running services.
- Only nodes in the **Available** or **Unavailable** state can be restarted.
- A single node can be in only one restart task at a time. Multiple restart tasks cannot be delivered to the same node at a time.
- If the node is being replaced, reset, or deleted, it cannot be restarted.
- When the driver of a resource pool is being upgraded, nodes in this resource pool cannot be restarted.
- After the node is restarted, it will be unavailable for a short time. The service is being started and the health status is being checked. Wait patiently.

Migrating a Node

Pay-per-use nodes can be migrated between resource pools.

To migrate nodes, the following requirements must be met:

- There must be at least one node left in the original resource pool after node migration.
- If the original resource pool has a logical subpool, the left resources in the original resource pool must meet the resource requirements of the logical subpool.
- There must be abundant available IP addresses in the target resource pool.
[Check the number of available IP addresses.](#)

To migrate a node, perform the following steps:

1. In the **Nodes** tab, select the target nodes and choose **More > Migrate Node** above the node list.
2. Click **Select Resource Pool** on the displayed page.

Figure 3-26 Selecting the target resource pool



3. In the **Select Resource Pool** dialog box, select the target resource pool to which the nodes are to be migrated and click **OK**.
4. Select the target node pool and click **Next: Confirm Specifications**.

Figure 3-27 Selecting the target node pool



5. Confirm the node migration information and click **Submit**.

Figure 3-28 Confirming the node migration information

Target Resource Pool

Name	Status	Available/Total Nodes	Description
	● Running	0/0	--

Target Node Pool

Name	Specifications	Step	Container Engine Space	Disk
				--

Migratable Nodes

Name	Node Status	Batch	Specifications	Driver Version	Driver Status	IP Address	Remarks
	● Available	--		--	--		

In the upper right corner of the node list, click **Records** to view the operation records of node migration in the current resource pool.

Deleting or Unsubscribing from a Node

- To release a single node from a pay-per-use resource pool, locate the target node and click **Delete** in the **Operation** column, enter **DELETE** in the displayed dialog box, and click **OK**.
To delete nodes in batches, select the target nodes and click **Delete** above the node list, enter **DELETE** in the displayed dialog box, and click **OK**.
- For a yearly/monthly resource pool whose resources are not expired, click **Unsubscribe** in the **Operation** column.
- For a yearly/monthly resource pool whose resources are expired (in the grace period), click **Release** in the **Operation** column.

If the delete button is available for a yearly/monthly node, click the button to delete the node.

NOTE

- Before deleting, unsubscribing from, or releasing a node, ensure that there are no running jobs on this node. Otherwise, the jobs will be interrupted.
- Delete, unsubscribe from, or release abnormal nodes in a resource pool and add new ones for substitution.
- If there is only one node, it cannot be deleted, unsubscribed from, or released.

Enabling/Disabling the Deletion Lock

To prevent nodes from being deleted or unsubscribed by mistake, you can enable the deletion lock. Once enabled, the nodes cannot be deleted or unsubscribed unless the lock is disabled.

 NOTE

- The deletion lock can be enabled only for the nodes in the resource pool.
- If the deletion lock is enabled, only node deletion and unsubscription are restricted. Other operations, such as node replacement, node restart, and node reset, work properly. Moreover, the resource pool that contains the nodes with deletion lock enabled can be deleted.
- Enabling deletion lock: Locate the target node and choose **More > Enable Deletion Lock** in the **Operation** column. In the displayed dialog box, confirm the information, enter **YES** in the text box, and click **OK**.
To enable deletion lock for multiple nodes in batches, select the target nodes and choose **More > Enable Deletion Lock** above the node list.
- Disabling deletion lock: Locate the target node and choose **More > Disable Deletion Lock** in the **Operation** column. In the displayed dialog box, confirm the information, enter **YES** in the text box, and click **OK**.
To disable deletion lock for multiple nodes in batches, select the target nodes and choose **More > Disable Deletion Lock** above the node list.

Authorizing O&M on the Event Center Page

To view the faulty nodes reported by the ModelArts O&M platform, log in to the ModelArts console. In the navigation pane on the left, choose **Event Center**. The planned events of the faulty nodes are displayed, including the basic information, event type, event status, and event description. You can either redeploy the nodes or authorize Huawei technical support to perform O&M operations.

- **Authorization conditions**

[Table 3-9](#) lists the event types and event status of the authorization operations that can be performed on the faulty node.

Table 3-9 Authorization conditions

Event Type	Event Status	Authorization Operations
System maintenance	Authorization Pending	Authorization and redeployment
Local disk recovery	Authorization Pending	Authorization and redeployment After the local disk is recovered, you can restore the partition by resetting the node . WARNING After authorization, recovering the local disk will cause local disk data loss. Therefore, migrate services and back up data before authorization.
Restarting a node	Authorization Pending	Authorization
O&M authorization	Authorization Pending	Authorization

Event Type	Event Status	Authorization Operations
Supernode maintenance	Authorization Pending	Authorization
Supernode redeployment	Authorization Pending	Redeployment Redeployment of supernodes must be performed within physical supernodes. When supernodes are sold out, redeployment is not supported, and the authorization button becomes unavailable.
Supernode local disk recovery	Authorization Pending	Authorization WARNING After authorization, recovering the local disk will cause local disk data loss. Therefore, migrate services and back up data before authorization.

- **Authorization**

If the faulty nodes meet the requirements listed in [Table 3-9](#), you can authorize Huawei technical support to perform O&M on the faulty nodes.

To do so, log in to the ModelArts console. In the navigation pane on the left, choose **Event Center**. Locate the target node and click **Authorize** in the **Operation** column. In the displayed dialog box, click **OK**.

If the planned event does not meet the requirements listed in [Table 3-9](#), the **Authorize** button becomes unavailable.

After the O&M, Huawei technical support will disable the authorization. No further operations are required.

- **Redeployment**

If the faulty nodes meet the redeployment requirements listed in [Table 3-9](#), you can authorize Huawei technical support to redeploy the faulty nodes.

After the O&M, Huawei technical support will disable the authorization. No further operations are required.



Redeploying nodes can restore them quickly, but local disk data will be lost. Therefore, migrate services and back up data before redeployment.

- To redeploy a node, log in to the ModelArts console. In the navigation pane on the left, choose **Event Center** under **Resource Management**, locate the node, and click **Redeploy** in the **Operation** column.
If the planned event does not meet the requirements listed in [Table 3-9](#), the **Redeploy** button becomes unavailable.
- Check whether **Forcible redeployment** is selected, enter **YES** in the text box, and click **OK**.

Redeployment depends on the node status. If the node is unavailable, redeployment cannot be completed. However, you can select **Forcible redeployment** to forcibly redeploy the node.

 WARNING

Forcible redeployment resets the node, deleting all data on both its local and cloud disks. Exercise caution when performing this operation.

FAQ

How do I locate a faulty node in a standard resource pool?

In a standard resource pool, ModelArts will add a taint to a faulty Kubernetes node so that jobs will not be scheduled to the tainted node. You can locate the fault by referring to the isolation code and detection method. For details, see [Faulty Nodes in a Standard Resource Pool](#).

3.3.5 Modifying the Job Types Supported by a Standard Dedicated Resource Pool

Description

ModelArts supports many types of jobs. Some of them can run in dedicated resource pools, including training jobs, inference services, and development environments.

You can change job types supported by a dedicated resource pool. Currently, training job, inference services, and development environments are supported.

Only selected types of jobs can be delivered to the corresponding dedicated resource pool.

To support different job types, different operations are performed in the backend, such as installing plug-ins and setting the network environment. Some operations use resources in the resource pool. As a result, available resources for you decrease. Therefore, select only the job types you need to avoid resource waste.

 WARNING

Modifying configured job types may cause the following problems: development environment instances are deleted, running training jobs fail, and running inference services fail.

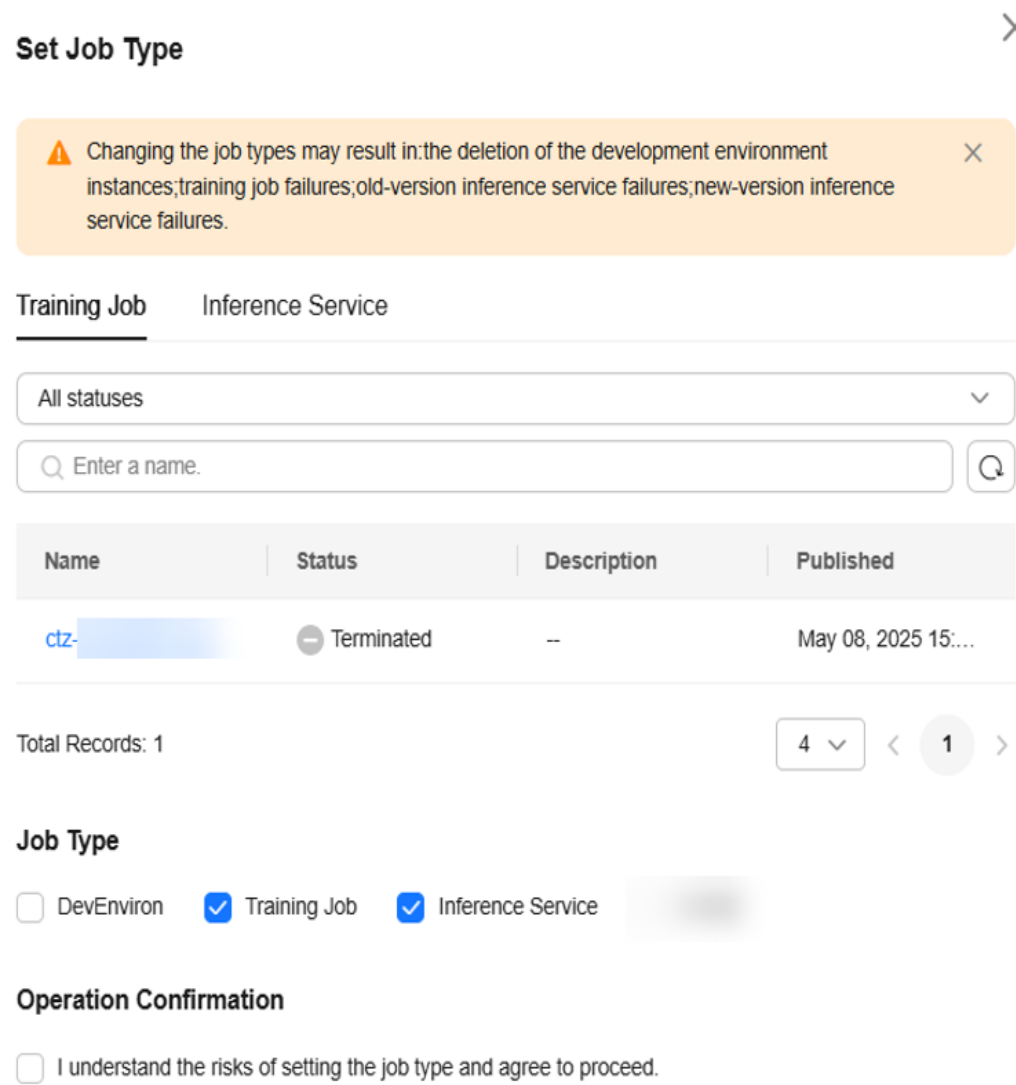
Constraints

The target dedicated resource pool must be running.

Procedure

1. Log in to the [ModelArts console](#). In the navigation pane on the left, choose **Standard Cluster** under **Resource Management**.
2. Locate the target resource pool in the list and choose **...** > **Set Job Type** in the **Operation** column.
3. In the **Set Job Type** dialog box, select job types.

Figure 3-29 Setting the job type



4. Read the risk warning, select it, and click **OK**.

3.3.6 Migrating Standard Dedicated Resource Pools and Networks to Other Workspaces

Background

The workspace of a dedicated resource pool is associated with an enterprise project, which involves bill collection. ModelArts provides workspaces to isolate resource operation permissions of different IAM users.

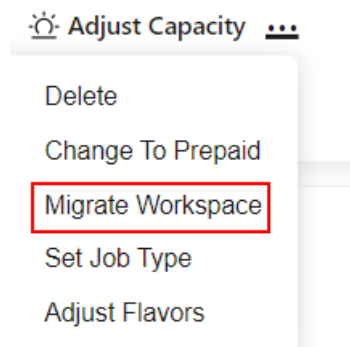
ModelArts allows you to migrate resource pools and networks to a target workspace. Once migrated, the resource pools will be associated with the workspace for you to manage and use. To ensure that resource pools are running properly in the target workspace, migrate the related networks together.

Constraints

- IAM users can only migrate resource pools and networks in their own workspaces.
- The resource pool must be running with no running jobs or tasks. Stop or migrate the running jobs if there are any.
- During the migration, do not perform other operations on the resource pool, such as scaling and job submission. Otherwise, the migration result may be affected.

Migrating the Workspace for a Resource Pool

1. Log in to the [ModelArts console](#). In the navigation pane on the left, choose **Standard Cluster** under **Resource Management**.
2. Locate the target resource pool in the list and choose **...** > **Migrate Workspace**.



3. In the displayed dialog box, select the target workspace, enter **YES**, and click **OK**.

Go to the target workspace. In the navigation pane on the left, choose **Standard Cluster** under **Resource Management**. If the resource pool is running in the target workspace, the migration is successful.

Migrating the Workspace for a Network

1. Log in to the [ModelArts console](#). In the navigation pane on the left, choose **Network** under **Resource Management**.
2. In the network list, choose **More** > **Migrate Workspace** in the **Operation** column of the target network.
3. In the displayed dialog box, select the target workspace and click **OK**.

Go to the target workspace. In the navigation pane on the left, choose **Network** under **Resource Management**. If the network is available in the target workspace, the migration is successful.

Figure 3-30 Migrating the workspace



FAQ

What will happen if the resource pool is migrated to another workspace yet the network is not migrated?

- Inconsistent network configurations: The resource pool will still use the network configurations of the original workspace when running in the target workspace. If the network environments of the original and target workspaces are different, the resource pool may fail to connect to the network resources in the target workspace.
- Network access issues: The resource pool may fail to access other resources or services in the target workspace when it is running in the target workspace.
- Security policy conflicts: The target workspace may have different security policies from the original workspace. In this case, the resource pool may fail to meet the new security policy requirements in the target workspace, resulting in restricted access or other security problems.
- Restricted job types: The job types supported by the resource pool, such as training jobs, inference services, and development environments, may depend on specific network configurations. If the network is not migrated, some job types may not run properly.

3.3.7 Configuring the Standard Dedicated Resource Pool to Access the Internet

Scenario

When you use a dedicated resource pool to create a job, for example, a training job, if the dedicated resource pool needs to access the Internet during job runtime, interconnect with VPC for VPC peering connection between the dedicated resource pool and the ECS bound with an EIP. Then, configure a public network NAT gateway for the ECS. In this way, the dedicated resource pool can access the Internet directly.

Billing

When configuring the public network of a standard resource pool, you need to purchase EIPs and public NAT gateways. EIPs are billed by bandwidth and reserved EIP fee. For details, see [EIP Pricing Details](#). NAT gateways are billed based on the gateway type, specifications, and running duration. For details, see [NAT Gateway Pricing Details](#).

Prerequisites

- You have obtained the ECS where the SNAT function is to be deployed.
- The ECS where the SNAT function is to be deployed runs the Linux OS.
- The ECS where the SNAT function is to be deployed has only one network interface card (NIC) configured.

Step 1: Interconnecting with the VPC

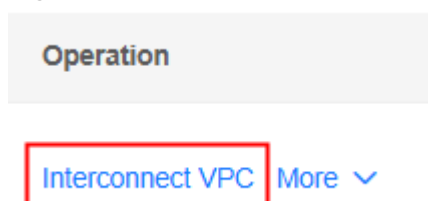
To interconnect with multiple VPCs is to establish network links between them or between a VPC and another network environment. This enables resource sharing and secure communication between VPCs.

With VPC interconnection, you can access resources across VPCs, share resources efficiently, improve data transfer speed, and maintain continuous services.

By interconnecting with VPCs, dedicated resource pools can communicate with ECSs that have EIPs bound.

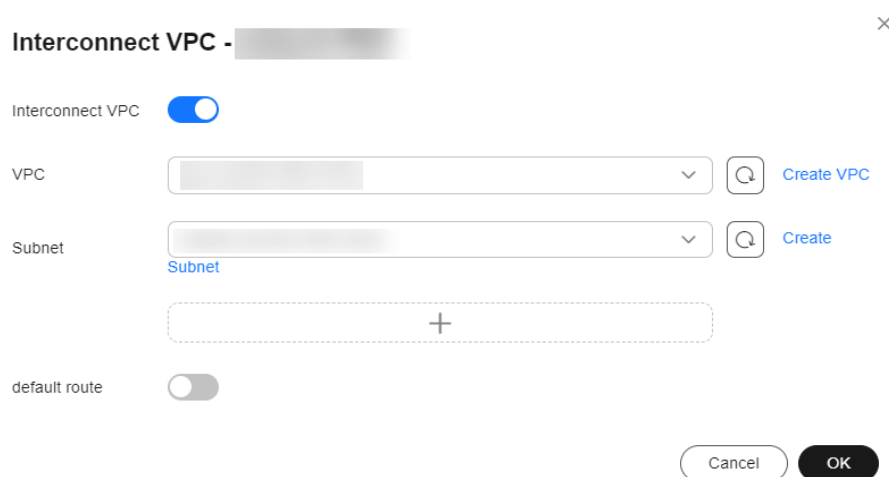
1. Log in to the [ModelArts console](#). In the navigation pane on the left, choose **Network** under **Resource Management**.
2. Click **Interconnect VPC** in the **Operation** column of the target network.

Figure 3-31 Interconnect VPC



3. In the displayed dialog box, click the button on the right of **Interconnect VPC**, and select an available VPC and subnet from the drop-down lists.
The peer network to be interconnected cannot overlap with the current CIDR block.

Figure 3-32 Parameters for interconnecting a VPC with a network



- If no VPC is available, click **Create VPC** on the right to create a VPC.
- If no subnet is available, click **Create Subnet** on the right to create a subnet.
- A VPC can interconnect with at most 10 subnets. To add a subnet, click the plus sign (+).
- To enable a dedicated resource pool to access the public network through a VPC, create a SNAT in the VPC, as the public network address is unknown. After the VPC is interconnected, by default, the public address cannot be forwarded to the SNAT of your VPC. Submit a service ticket and contact technical support to add a default route. Then, when you interconnect with a VPC, ModelArts **0.0.0.0/0** is used as the default route. In this case, you do not need to submit a service ticket. Add the default route for network configuration.

Step 2: Configuring a Public SNAT Gateway

Bind an EIP to an ECS, configure a public network NAT gateway on the ECS that has been bound to an EIP, and use the SNAT rule of the public network NAT gateway to access the Internet.

Configure and verify SNAT by referring to [Using a Public NAT Gateway to Enable Servers to Share One or More EIPs to Access the Internet](#).

1. Buy an EIP.
 - a. Go to the [Buy EIP](#) page.
 - b. On the **Buy EIP** page, set the EIP name to **EIP-A**.
Configure other EIP parameters as required. For details, see [Assigning an EIP](#).
 - c. Configure the parameters and click **Next**.
Return to the EIP list to view **EIP-A** you have assigned.
2. Buy a public NAT gateway.
 - a. Go to the [Buy Public NAT Gateway](#) page.
 - b. On the **Buy Public NAT Gateway** page, configure required parameters.

Table 3-10 Parameters for configuring a public NAT gateway

Parameter	Example	Description
Region	CN North-Beijing4	The region where the public NAT gateway is located.
Billing Mode	Pay-per-use	The billing mode of the public NAT gateway.
Specifications	Small	The specifications of the public NAT gateway. The value can be Extra-large , Large , Medium , or Small . For details about specifications, click Learn more on the page.

Parameter	Example	Description
Name	public-nat-01	Name of the public NAT gateway. Enter up to 64 characters. Only letters, digits, underscores (_), hyphens (-), and periods (.) are allowed.
VPC	VPC-A	The VPC that the public NAT gateway belongs to. The selected VPC cannot be changed after the public NAT gateway is purchased. NOTE To allow traffic to pass through the public NAT gateway, a route to the public NAT gateway in the VPC is required. When you buy a public NAT gateway, a default route 0.0.0.0/0 to the public NAT gateway is automatically added to the default route table of the VPC. If the default route 0.0.0.0/0 already exists in the default route table of the VPC before you buy the public NAT gateway, the default route that points to the public NAT gateway will fail to be added automatically. In this case, perform the following operations after the public NAT gateway is bought: Manually add a different route that points to the gateway or create a default route 0.0.0.0/0 pointing to the gateway in the new routing table.
Subnet	Subnet-A01	The subnet that the public NAT gateway belongs to. The subnet must have at least one available IP address. The selected subnet cannot be changed after the public NAT gateway is purchased. The NAT gateway will be deployed in the selected subnet. The NAT gateway works for the entire VPC where it is deployed. To enable communications over the Internet, add SNAT or DNAT rules.
(Optional) Advanced Settings	-	Click the drop-down arrow to configure advanced parameters of the public NAT gateway.
SNAT Connection TCP Timeout (s)	900	The timeout period of a TCP connection established using the SNAT rule. If no data is exchanged within this period, the TCP connection will be closed. Value range: 40 to 7200

Parameter	Example	Description
SNAT Connection UDP Timeout (s)	300	The timeout period of a UDP connection established using the SNAT rule. If no data is exchanged within this period, the UDP connection will be closed. Value range: 40 to 7200
SNAT Connection ICMP Timeout (s)	10	The timeout period of an ICMP connection established using the SNAT rule. If no data is exchanged within this period, the ICMP connection will be closed. Value range: 10 to 7200
TCP TIME_WAIT (s)	5	How long the side that actively closed the TCP connection is in the TIME_WAIT state. Value range: 0 to 1800
Description	Not required	Supplementary information about the public NAT gateway. Enter up to 255 characters. Angle brackets (<>) are not allowed.
Tag	Not required	The identifier of the public NAT gateway. A tag is a key-value pair. You can add up to 20 tags to each NAT gateway.

- c. Click **Next**. On the displayed page, confirm the public NAT gateway specifications.
- d. If you do not need to modify the information, click **Submit**.

On the **Public NAT Gateways** page, you can view the created public NAT gateway in the list.

3. Add an SNAT rule.
 - a. On the displayed page, click the name of the public NAT gateway on which you need to add an SNAT rule.
 - b. In the **SNAT Rules** tab, click **Add SNAT Rule**.
 - c. Configure the required parameters. [Table 3-11](#) describes the parameters.

Table 3-11 SNAT parameters

Parameter	Example	Description
Scenario	Direct Connect/ Cloud Connect	If SNAT is used to access the Internet, select Direct Connect or Cloud Connect . The value indicates that the server in the local data center of Direct Connect or VPN uses the SNAT rule to access the Internet.

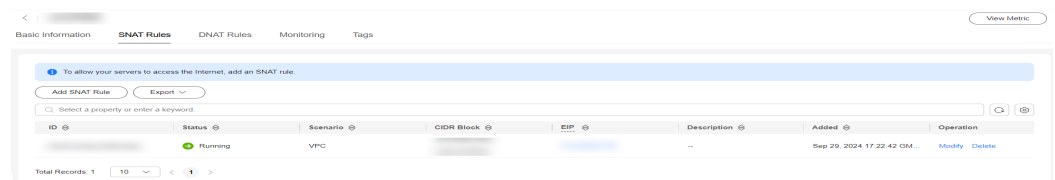
Parameter	Example	Description
CIDR Block	Existing	On-premises servers whose IP address in this CIDR block can access the Internet through the SNAT rule. Set this parameter to the default ModelArts dedicated resource pool CIDR block 192.168.20.0/24. Select a CIDR block from the drop-down list.
Public IP Address Type	EIP	Used to access the Internet.
Monitor	-	Monitoring of the number of SNAT connections. You can set alarm rules to monitor your SNAT connections and keep informed of any changes in a timely manner.
Description	Not required	Supplementary information about the SNAT rule. Enter up to 255 characters. Angle brackets (<>) are not allowed.

d. Click **OK**.

4. In the **SNAT Rules** tab, view details of the SNAT rule.

If **Status** of the SNAT rule is **Running**, the SNAT rule has been created.

Figure 3-33 Verifying that the SNAT rule has been added



Step 3: Adding Routes for VPC Peering Connection

After VPC is interconnected, jobs in the dedicated resource pool cannot be forwarded to the SNAT of the user VPC by default. You need to add a default route for the VPC peering connection on ModelArts.

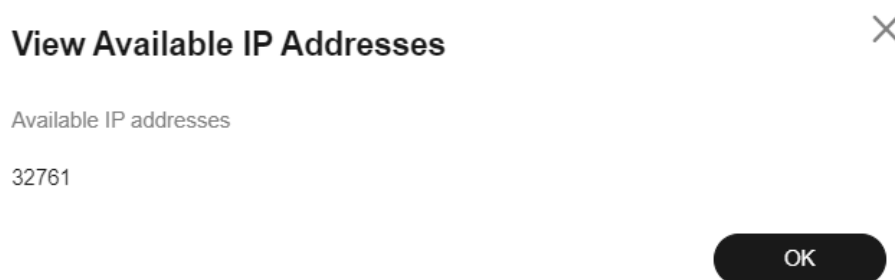
If the default route is not enabled in [Step 1: Interconnecting with the VPC](#), submit a service ticket to contact technical support to add a default route for VPC peering connection.

If the default route is enabled in [Step 1: Interconnecting with the VPC](#), when you interconnect with a VPC, ModelArts **0.0.0.0/0** is used as the default route. In this case, you do not need to submit a service ticket. Add the default route for network configuration.

(Optional) Checking the Number of Available IP Addresses

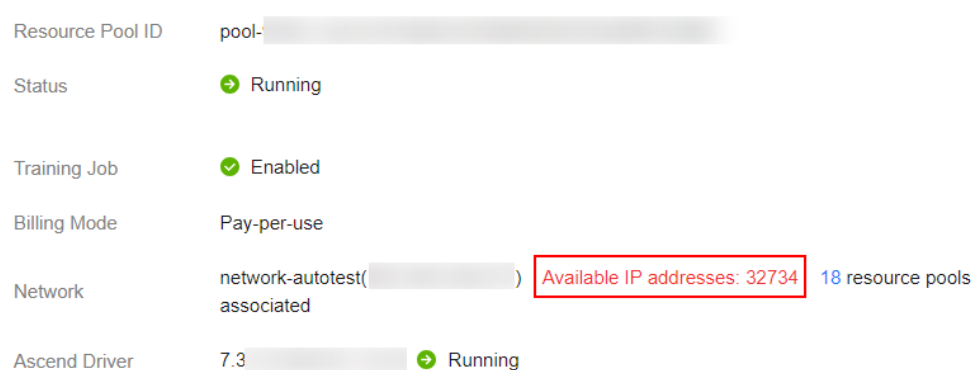
Log in to the [ModelArts console](#). In the navigation pane on the left, choose **Network**. Then, locate the target network and choose **More > View Available IP Addresses** in the **Operation** column.

Figure 3-34 Viewing available IP addresses



On the resource pool details page, you can view the number of available IP addresses bound to the network.

Figure 3-35 Viewing available IP addresses



FAQ

How Do I Add a VPC Peering Connection Between a Dedicated Resource Pool and an SFS in ModelArts?

Interconnect a VPC with a ModelArts resource pool so that the resource pool and SFS share the same VPC. When you create a training job, the SFS option will be available.

For details about how to interconnect with a VPC, see [Interconnecting a VPC with a ModelArts Network](#).

3.3.8 Using TMS Tags to Manage Resources by Group

ModelArts can work with Tag Management Service (TMS). When creating resource-consuming tasks in ModelArts, configure tags for these tasks so that ModelArts can use tags to manage resources by group.

ModelArts allows you to configure tags when you create training jobs, notebook instances, real-time inference services, or standard dedicated resource pools.

Operation Process

1. [Step 1 Create Predefined Tags on TMS](#)
2. [Step 2 Add a Tag to a ModelArts Task](#)
3. [Step 3 Obtain ModelArts Resource Usage by Resource Type in TMS](#)

Step 1 Create Predefined Tags on TMS

Log in to the [TMS console](#) and create a tag on the **Predefined Tags** page. The created tags are global and can be used in all Huawei Cloud regions.

Step 2 Add a Tag to a ModelArts Task

When creating a notebook instance, training job, or real-time inference services in ModelArts, configure a tag for the task.

- Add a tag to a ModelArts notebook instance.
Add a tag when you create a notebook instance. Alternatively, after creating a notebook instance, add a tag on the **Tags** tab on the instance details page.
- Add a tag to a ModelArts training job.
Add a tag when you create a training job. Alternatively, after creating a training job, add a tag on the **Tags** tab on the job details page.
- Add a tag to a ModelArts real-time service.
Add a tag when you create a real-time service. Alternatively, after creating a real-time service, add a tag on the **Tags** tab on the service details page.
- Add a tag to a ModelArts dedicated resource pool.
Add a tag when you create a standard dedicated resource pool. Alternatively, after creating a dedicated resource pool, add a tag in the **Tags** tab on the resource pool details page.

Figure 3-36 Adding a Label

Add Tag

It is recommended that you use TMS's predefined tag function to add the same tag to different cloud resources. [View predefined tags](#)

To add a tag, enter a tag key and a tag value below.

key = value

Enter a tag key Enter a tag value Add

19 tags available for addition.

Cancel OK

When adding a tag to a ModelArts task, you can create new tags by specifying the keys and values of the new tags. The tags created here are available only to the current project.

Step 3 Obtain ModelArts Resource Usage by Resource Type in TMS

Log in to the [TMS console](#) and search for resource tasks in a specified region by resource type and resource tag on the **Tag Management** page.

- **Region:** For details about Huawei Cloud regions, see [Region and AZ](#).
- **Resource Type:** [Table 3-12](#) lists the resource types that can be viewed on ModelArts.
- **Resource Tag:** If no tag is specified, all resources are displayed, regardless of whether the resources are configured with tags. One or multiple tags can be selected to obtain resource usage.

Table 3-12 Resource types that can be viewed on ModelArts

Resource Type	Description
ModelArts-Notebook	Notebook instances in ModelArts DevEnviron
ModelArts-TrainingJob	ModelArts training jobs
ModelArts-RealtimeService	ModelArts real-time inference services
ModelArts-ResourcePool	ModelArts dedicated resource pools

If your organization has configured tag policies for ModelArts, add tags to resources based on the policies. If a tag does not comply with the tag policies,

resource creation may fail. Contact your organization administrator to learn more about tag policies.

3.3.9 Managing Logical Subpools of a Standard Dedicated Resource Pool

You can split logical subpools from a ModelArts Standard dedicated resource pool and set resource quotas for each subpool, including reserved quotas and quota rollout. After a logical subpool is split, tasks in the training, inference, and development environments must be delivered to the logical subpool and restricted by the quota. Only tasks within the quota can be delivered and run.

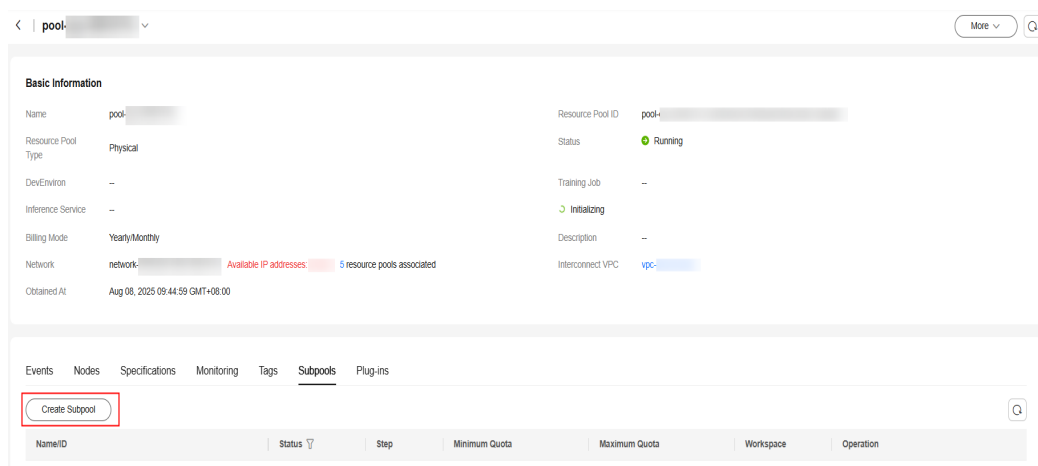
On the resource pool details page, click **Subpools**. In the logical subpool list, you can click **Create Subpool** in the upper left corner to create a logical subpool. You can also scale in or out a created logical subpool, set the job types, and delete the logical subpool.

If you need to migrate the resources of one logical subpool to another pool, delete one subpool, and scale out the other subpool.

Creating a Logical Subpool

1. Log in to the [ModelArts console](#). In the navigation pane on the left, choose **Standard Cluster** under **Resource Management**.
2. Click the target resource pool name to access its details page.
3. On the resource pool details page, click **Subpools**. Then, click **Create Subpool** in the upper left corner.

Figure 3-37 Logical subpool



4. Click **Create Subpool**. In the displayed dialog box, set the parameters and click **OK**.

If a created logical subpool needs to be bound to a physical pool node, [enable node binding](#).

Table 3-13 Parameters for creating a logical subpool

Parameter	Description
Name	Name of a logical pool.
Description	Description of a logical subpool.
Job Type	Job type supported by the logical subpool. The job type must be supported by the physical dedicated resource pool.
Instance Specification Management	Set the following parameters. ModelArts will schedule tasks for the logical subpool based on the configured parameters. • Step: the smallest unit for changing the minimum quota. In node binding, nodes at each step are treated as one group. Step is available only for supernodes and full-rack nodes. In the supernode scenario, the step size of a logical subpool can be smaller than that of a physical pool. The step size cannot be changed after a logical subpool is created. Set a proper step size as required. • Minimum Quota: minimum resources of a logical subpool. The value can be 0. • Maximum Quota: maximum resources of a logical subpool. The value must be no less than the guaranteed quota. If the minimum quota is less than the maximum quota, Development Environment cannot be selected for Job Type. Task scheduling rules: • If the required resources are no more than the minimum quota, the resource usage can be guaranteed. • If the required resources are more than the minimum quota, and no more than the maximum quota, the jobs can be scheduled when the resources are sufficient and will be preempted when the resources are insufficient.

Figure 3-38 Creating a Logical Subpool

Create Subpool

Name

pool-6292

Description (Optional)

Enter a description.

0/100

Workspace

default

Job Type

☒ Inference Service

Instance Specification Management

Flavor	Step	Minimum Quota	Maximum Quota
☒ modelarts.vm.8u...	--	Please Enter	Please Enter

Scaling a Logical Subpool

Locate the target logical subpool and click **Adjust Capacity** in the **Operation** column. In the displayed dialog box, reset the instance specifications and click **OK**.

Figure 3-39 Scaling a logical subpool

Resize

Name

pool-3f34

Description

0/100

Workspace

default

Job Type

☒ Training Job

Instance Specification Management

Flavor	Step	Minimum Qu...	Maximum Qu...
☒ modelarts.bm.npu.arm.8snt9b3...	--	1	1

Association Configuration

☐ Node Draining

Once enabled, added or removed nodes will be drained. **This function only applies to the current operation.** During drainage, nodes will be unschedulable, and job workloads will be evicted, potentially causing job failures.

Cancel

OK

For logical subpools with node binding enabled, check whether node draining needs to be enabled during scaling. Node draining is to clear node resources to release or remove nodes that are no longer needed.

If node draining is enabled, the drainage will be performed on the nodes to be scaled only this time. You can check the drainage status in the **Nodes** tab. If the draining fails, hover the cursor over the node status to view the failure cause.

WARNING

During the draining, nodes cannot be scheduled and job loads on the nodes will be evicted, which may cause job failures.

Setting the Job Type of a Logical Subpool

Locate the target logical subpool and click **Set Job Type** in the **Operation** column. In the displayed dialog box, set the target job type and click **OK**.

WARNING

Modifying configured job types may cause the following problems: development environment instances are deleted, running training jobs fail, and running inference services fail.

Deleting a Logical Subpool

Locate the target logical subpool and click **Delete** in the **Operation** column. In the displayed dialog box, enter **DELETE** and click **OK**.

Figure 3-40 Deleting a logical subpool

Delete Dedicated Resource Pool

X

The following resource pool will be deleted.

Pool Name	Resource Pool ID	Resource Pool Type	Used By
		Subpools	Training Jobs Inf...

To confirm deletion, enter "DELETE" below. [Auto Enter](#)

DELETE

Enabling/Disabling Node Binding for a Standard Dedicated Resource Pool

If a logical subpool has been created in a resource pool, you can enable node binding to bind dedicated nodes to the logical subpool.

Once enabled, the system will automatically bind the corresponding number of dedicated nodes. Tasks can only run on the bound nodes to ensure resource

dedication and stability. This function applies to tasks that require high resource assurance and stability.

Node binding can:

- Improve task stability. By binding specific nodes, tasks will not be interrupted due to dynamic node allocation, ensuring task continuity and stability.
- Assure resources. After node binding is enabled, tasks can exclusively use the bound node resources, preventing resource competition with other tasks.
- Optimize resource usage. Proper allocation of node resources can improve overall resource usage and avoid resource waste.

Enabling node binding

After node binding is enabled, the service background automatically binds nodes to the logical subpool.

Locate the target resource pool in the list and choose **...** > **Enable Node Binding** on the right.

In the displayed **Enable Node Binding** dialog box, click **OK**.

1. Log in to the [ModelArts console](#). In the navigation pane on the left, choose **Standard Cluster** under **Resource Management**.
2. Locate the target resource pool in the list and choose **...** > **Enable Node Binding** on the right.
3. In the displayed **Enable Node Binding** dialog box, click **OK**.

After node binding is enabled for the standard dedicated resource pool, nodes in a resource pool can be [bound to logical subpools](#).

Disabling node binding



Once node binding is disabled, the jobs in the original logical subpool may be occupied. Exercise caution when performing this operation.

Locate the target resource pool in the list and choose **...** > **Disable Node Binding** on the right.

1. In the displayed dialog box, confirm the information, enter **YES** in the text box, and click **OK**.
2. Log in to the [ModelArts console](#). In the navigation pane on the left, choose **Standard Cluster** under **Resource Management**.
3. Locate the target resource pool in the list and choose **...** > **Disable Node Binding** on the right.
4. In the displayed dialog box, confirm the information, enter **YES** in the text box, and click **OK**.

Binding to Another Logical Subpool

After **node binding is enabled** for the standard dedicated resource pool, nodes in a resource pool can be bound to a specified logical subpool.

1. Log in to the **ModelArts console**. In the navigation pane on the left, choose **Standard Cluster** under **Resource Management**.
2. Click the resource pool with node binding enabled. The resource pool details page is displayed.
3. Switch to the **Nodes** tab, select the target nodes, and click **Bind Logical Subpool** above the list.
4. In the **Bind Logical Subpool** dialog box, set the parameters for binding the nodes. The nodes that meet the parameter settings can be bound to the logical subpool.



WARNING

- The target nodes must be in a steady (**Available** or **Abnormal**) state.
- If the driver version of the logical subpool does not match the nodes, the node driver will automatically be upgraded, which will interrupt services.

Table 3-14 Parameters

Parameter	Description
Node Source	Select the resource pool type of the node before the binding. You can select a physical resource pool or a logical subpool.
Node Type	Select the node type of the node to be bound. You can select a single node or a supernode. • Single node: A physical or virtual server that provides independent basic compute, storage, and network resources • Supernode: A converged node that provides a large-scale compute resource pool and supports flexible allocation and high-density deployment. Supernodes are specially designed to support large-scale model inference tasks. These servers are usually equipped with multiple compute cards (such as Ascend NPUs) to provide powerful computing capabilities to meet heavy-load inference requirements. Supernodes are supported only in CN Southwest-Guiyang1 and CN East2.
Instance Specifications	Select instance specifications for the node. You must select nodes with identical specifications to rebind them to a new resource pool.

Parameter	Description
Node Batch	If Node Type is set to Supernode , you need to select the batch of nodes to be rebound. You must select nodes of the same batch to rebound them to a new resource pool.
Node List	Select the target nodes in the list.
Drain Nodes	Node draining is to clear node resources to release or remove nodes that are no longer needed. After node draining is enabled, nodes in the logical subpool will be drained. The draining operation takes effect only for the current logical subpool rebinding. You can check the drainage information in the Nodes tab. If the draining fails, hover the cursor over the node status to view the failure cause. WARNING During the draining, nodes cannot be scheduled and job loads on the nodes will be evicted, which may cause job failures.

- Click **Next**, select the target resource pool type and the target resource pool, and click **OK**.

If **Node Source** is **Physical resource pool**, the target resource pool must be a logical subpool. If **Node Source** is **Logical subpool**, the target resource pool can be a physical resource pool or a logical subpool.

If **Node Source** is **Supernode**, the node can be bound only to a logical subpool that matches the instance specifications and the selected nodes can be divided by the resource pool specification step.

3.3.10 Releasing Standard Dedicated Resource Pools and Deleting the Network

The service is complete and resources have been idle for a long time. If there are no running tasks in the resource pool and data has been backed up and migrated, you can delete the dedicated resource pool and network to release resources.

Precautions

- Before releasing the resources, ensure that all important data has been backed up to prevent data loss caused by misoperations.
- If there are running jobs or tasks in the resource pool, stop and migrate them first.

Deleting a Resource Pool

If a dedicated resource pool is no longer needed for AI service development, you can delete the resource pool to release resources.

 WARNING

- Deleting a resource pool will delete the associated resources as well. The operation cannot be undone.
 - If there are nodes with deletion lock enabled in the resource pool, such nodes will be deleted when the resource pool is deleted, which interrupts running services. Exercise caution when performing this operation as it cannot be rolled back and ensure that key services are not affected.
-

1. Log in to the [ModelArts console](#). In the navigation pane on the left, choose **Standard Cluster** under **Resource Management**.
2. Locate the target resource pool in the list and choose **...** > **Delete** in the **Operation** column.

 **Adjust Capacity** **...**

Delete

3. On the displayed page, check the associated resources and jobs that will be affected, enter **DELETE** in the text box, and click **OK**.
For yearly/monthly nodes that are not unsubscribed from or released, go to the resource pool details page and unsubscribe from or release the nodes. For details, see [Viewing Resource Pool Nodes](#).
After the resources are released, the resource pool will be removed from the list.

Figure 3-41 Deleting a resource pool

×

Delete Dedicated Resource Pool

The following resource pool will be deleted.

Pool Name	Resource Pool ID	Resource Pool Type	Used By
pool-		Physical	Inference Service ...

Associated Resources

⚠ 1. Deleting a resource pool will delete the associated resources as well. The operation cannot be undone.

2. You can retain some plug-ins. The retained plug-ins will stay in the Kubernetes cluster.

▽ Nodes (6)

▽ Plug-ins (4)

Associated Jobs

⚠ Deleting a resource pool will affect the jobs running on the resource pool.

▽ Real-Time Inference (9)

To confirm deletion, enter "**DELETE**" below. [Auto Enter](#)

DELETE

Deleting a Network

If a network is no longer needed for AI service development, you can delete the network.

1. Go to the **Network** tab, locate the target network in the list, and choose **More > Delete** in the **Operation** column.
2. Enter **DELETE** in the text box, confirm the information, and click **OK**.
The network will be removed from the network list once deleted.

FAQ

Why there are deleted dedicated resource pools that failed to be created on the ModelArts console?

After a dedicated resource pool is deleted on the console, the backend releases the resources used by the pool. It takes several minutes to release the resources, during which the pool is still displayed on the console. To create a dedicated resource pool again, wait 5 minutes after the deletion. Additionally, do not use the

name of the dedicated resource pool that fails to be created to name the new dedicated resource pool. To perform an automated test on the UI, it is a good practice to use a random string as the name of the created dedicated resource pool.

3.4 Managing Standard Dedicated Resource Pool Plug-ins

3.4.1 Overview

ModelArts provides various plug-ins for you to selectively expand resource pool functions to meet service requirements.

Plug-ins Installed by Default

When you create a dedicated resource pool, certain plug-ins will be installed by default based on the resource pool type and job type.



Plug-ins installed by default in a resource pool cannot be uninstalled.

Table 3-15 Plug-ins installed by default

Plug-in	Description
Node Fault Detection (ModelArts Node Agent)	ModelArts Node Agent is a plug-in for monitoring cluster node exceptions, also, a component for connecting to third-party monitoring platforms. It is a daemon that runs on each node to collect node problems from different daemon processes.
ModelArts Metric Collector	The metric-collector, a default built-in plug-in of ModelArts, runs as a node daemon to collect node and job metrics and report them to AOM.
AI Suite (ModelArts Device Plugin)	The AI suite, Ascend NPU, is a device management plug-in that supports Huawei NPUs in containers. When you create a dedicated resource pool , this plug-in is automatically installed only when the instance specification type is set to Ascend.

Plug-in	Description
Volcano Scheduler	Volcano is a batch scheduling platform based on Kubernetes. It provides a series of features required by machine learning, deep learning, bioinformatics, genomics, and other big data applications, as a powerful supplement to Kubernetes capabilities. When you create a dedicated resource pool, this plug-in is automatically installed when Job Type is set to Training Jobs.

Installing the Plug-in Manually

You can install plug-ins to extend resource pool functions as required.

Table 3-16 Plug-ins installed manually

Plug-in	Description
NodeLocal DNSCache	NodeLocal DNSCache is a set of daemons running on cluster nodes. It improves cluster DNS performance by using DNS cache proxies.
Cloud Native Log Collection	Log Collect is used to collect cloud native logs and Kubernetes events based on open-source fluent-bit and opentelemetry. log-agent supports CRD-based log collection policies. Based on your configured policy rules, it can collect container standard output logs, container file logs, node logs, and Kubernetes event logs in a cluster and forward them to Log Tank Service (LTS). It also reports all abnormal Kubernetes events and some normal Kubernetes events to Application Operations Management (AOM) for event alarm configuration.

Plug-in Lifecycle

Table 3-17 Plug-in lifecycle

Status	Status Attribute	Description
Installing	Intermediate	The plug-in is being deployed. If all instances cannot be scheduled due to incorrect plug-in configuration or insufficient resources, the system sets the plug-in status to Unavailable 10 minutes later.
Running	Stable	The plug-in is running, all plug-in instances are deployed, and the plug-in can be used properly.
Upgrading	Intermediate	The plug-in is being upgraded.
Not available	Stable	The plug-in is abnormal and cannot be used. You can click the status to view the failure cause.
Deleting	Intermediate	The plug-in is being deleted. If this state stays for a long time, an exception occurred.

Searching for a Plug-in on the Plug-in Square

The **ModelArts** Plug-in Square provides various plug-ins. You can view the plug-in details and install them to a specified resource pool as needed.

Table 3-18 Supported operations on the Plug-in Square

Operation	Description	Procedure
Searching for and viewing a plug-in	Search for and view a plug-in.	Log in to the ModelArts console . In the navigation pane on the left, choose Add-ons . Choose a resource type from the drop-down list to filter plug-ins, or enter a keyword in the search box to search for a plug-in.
Viewing plug-in details	View the plug-in details, including the plug-in introduction and component list.	1. Log in to the ModelArts console . In the navigation pane on the left, choose Add-ons . 2. Click the plug-in name to view its details.

Operation	Description	Procedure
Installing a plug-in	Certain plug-ins can be manually installed.	1. Log in to the ModelArts console. In the navigation pane on the left, choose Add-ons. 1. Locate the target plug-in and click Install. 2. In the displayed dialog box, select the resource type of the plug-in to be installed. For some plug-ins, you also need to select a plug-in version. Set the information and click Next. • Dedicated cluster: Install the plug-in to a resource pool. The supported resource pool types vary depending on the plug-in. See the supported types on the GUI accordingly. • Dedicated node: Install the plug-in to a specific node in the resource pool. Perform operations and run commands as prompted. 3. Configure related parameters. The configurations vary depending on the plug-in. For details, see section "Plug-ins".

Viewing the Plug-ins of a Standard Resource Pool on the Resource Pool Details Page

In the **Plug-ins** tab of the resource pool details page, perform the operations described in [Table 3-19](#).

Table 3-19 Related operations

Operation	Description	Procedure
Querying the plug-ins	View all plug-ins of a resource pool. On this page, you can view plug-in details, install, upgrade, and uninstall plug-ins, and manage plug-ins in a centralized manner.	1. Log in to the ModelArts console. In the navigation pane on the left, choose Standard Cluster under Resource Management. 2. Click the resource pool to access its details page. 3. Click the Plug-ins tab.
Viewing plug-in details	View the plug-in details, including the plug-in introduction and component list.	1. Log in to the ModelArts console. In the navigation pane on the left, choose Standard Cluster under Resource Management. 2. Click the resource pool to access its details page. 3. Click the Plug-ins tab. 4. Click the plug-in name to view its details.
Plug-ins installed by default	When you create a resource pool, certain plug-ins are installed by default.	Creating a Standard Dedicated Resource Pool

Operation	Description	Procedure
Installing the plug-in manually	You can manually install a plug-in in a resource pool.	Method 1: Install the plug-in when you create a standard dedicated resource pool. For details, see Creating a Standard Dedicated Resource Pool. Method 2: Install the plug-in on the Plug-in Square. For details, see Searching for a Plug-in on the Plug-in Square. Method 3: - Log in to the ModelArts console. In the navigation pane on the left, choose Standard Cluster under Resource Management. - Click the resource pool name to access its details page. - Click the Plug-ins tab. - Locate the plug-in to be installed and click Install. - In the displayed dialog box, configure the parameters. The configurations vary depending on plug-ins. For details, see "Plug-ins".
Editing a plug-in	Edit plug-in parameters.	- On the resource pool details page, click the Plug-ins tab. - Locate the plug-in to be edited and click Edit. The configurations vary depending on plug-ins. For details, see the configuration description of the plug-ins. Only the following plug-in versions can be edited: - ModelArts Node Agent 7.2.0 or later - AI suite (Ascend NPU) 2.1.53 or later - Volcano Scheduler 1.17.11 or later - Click OK.

Operation	Description	Procedure
Upgrading a plug-in	Upgrade the plug-in to the latest version.	1. On the resource pool details page, click the Plug-ins tab. 2. Locate the plug-in to be upgraded and click Upgrade. The configurations vary depending on plug-ins. For details, see the configuration description of the plug-ins. • Table 3-31 • Table 3-28 3. Click OK.
Uninstalling a plug-in	Uninstall a plug-in from the resource pool. This operation cannot be undone.	1. On the resource pool details page, click the Plug-ins tab. 2. Locate the plug-in to be uninstalled and click Uninstall. 3. In the displayed dialog box, enter DELETE and click OK.

Figure 3-42 Plug-in details

Plug-in Details

 **ModelArts Node Agent** 7.0.0-2...
Running

Introduction

Component List

Function

The ModelArts Node Agent is a plugin that monitors abnormal events on cluster nodes and a component that interfaces with third-party monitoring platforms. It runs as a daemon on each node, collecting node issues from various daemons.

Created

May 09, 2025 15:42:42 GMT+08:00

Version

7.0.0-20250414160729

Change History

Q

⚙

Version	New Feature
7.0.0-20250414160729	Kubernetes v1.31 is supported
7.0.0-20250409171907	Kubernetes v1.31 is supported
7.0.0-20250407150406	Kubernetes v1.31 is supported
7.0.0-20250331170107	Kubernetes v1.31 is supported
7.0.0-20250331145906	Kubernetes v1.31 is supported
7.0.0-20250327171246	Kubernetes v1.31 is supported
7.0.0-20250321143651	Kubernetes v1.31 is supported
7.0.0-20250314141623	Kubernetes v1.31 is supported
7.0.0-20250313194413	Kubernetes v1.31 is supported
7.0.0-20250313101711	Kubernetes v1.31 is supported

Figure 3-43 Uninstalling a plug-in

 **Cloud Native Log Collection** 1.7.3
Running

Upgrade

Edit

Uninstall

FAQ

- If a mandatory plug-in is unavailable, or is being installed or deleted for a long time, contact ModelArts technical support.
- If a plug-in that can be installed or uninstalled is unavailable or is being installed or deleted for a long time, you can uninstall the plug-in and reinstall it. If the fault persists, contact ModelArts technical support.

3.4.2 Node Fault Detection (ModelArts Node Agent)

Description

ModelArts Node Agent is a plug-in for monitoring cluster node exceptions, also, a component for connecting to third-party monitoring platforms. It is installed in each Kubernetes resource pool by default. It is a daemon running on each node and can collect node problems from different daemon processes.

Installing a Plug-in

Certain plug-ins are automatically installed when you create a dedicated resource pool. Currently, you cannot upgrade the plug-ins yourself.

Components

Table 3-20 Plug-in component

Component	Description	Resource Type
maos-node-agent	Monitor cluster node exceptions and interconnect with third-party monitoring platforms.	DeamonSet

Change History

Table 3-21 Release history

Plug-in Version	New Feature
6.8.0	Added components for detecting ModelArts node faults and monitoring cluster node exceptions.
6.7.0	Added components for detecting ModelArts node faults and monitoring cluster node exceptions.
6.6.0	Added components for detecting ModelArts node faults and monitoring cluster node exceptions.

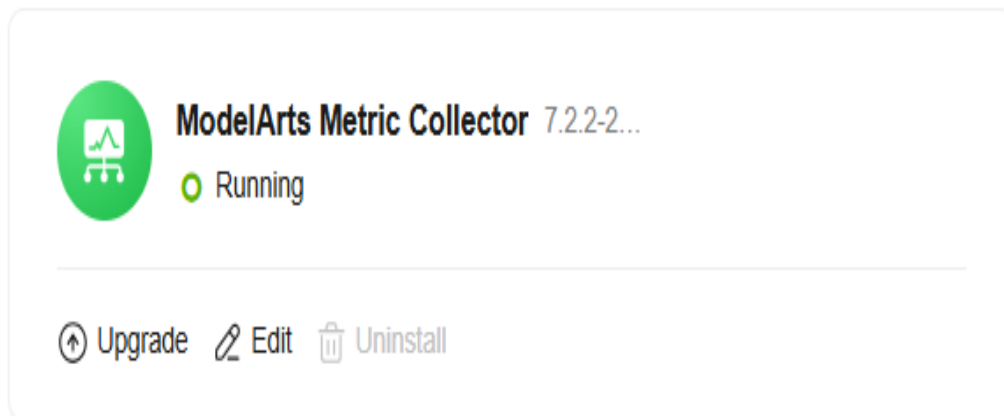
3.4.3 ModelArts Metric Collector

Description

Metric Collector, a default built-in plug-in of ModelArts, runs as a node daemon to collect node and job metrics and report them to AOM. For details about the

metric list, see [Viewing All ModelArts Monitoring Metrics on the AOM Console](#).

Figure 3-44 Metrics Collector



Constraints

- The plug-in is automatically installed during resource pool creation and cannot be uninstalled.
- This plug-in is automatically installed if ModelArts Node Agent is upgraded to the latest version for an existing resource pool.
- During the plug-in upgrade, the pod for metric collection restarts. As a result, metrics may not be reported for a short period of time. Exercise caution when performing the operation.

Components

Component	Description	Resource Type
modelarts-metric-collector	Node and container metrics collection	DaemonSet

Parameters

Parameter	Description
Standby Node Metric Reporting	Whether the standby node of a dedicated pool reports metrics. The default value is false .

Parameter	Description
Enable Exporter	Third-party monitoring systems such as Prometheus can obtain metrics collected by ModelArts. If this function is disabled, third-party monitoring systems such as Prometheus cannot collect metrics. This function is enabled by default. Dedicated pool: Enable this function if you want to use inference job metrics for scaling.
Report Metrics to a Custom Common Prometheus Instance on AOM	By default, metrics are reported to the Prometheus_AOM_Default instance of AOM. If this function is enabled, metrics are reported to the custom Prometheus common instance, as shown in Figure 3-45 . If this function is disabled, metrics are reported to the default Prometheus instance, that is, the Prometheus_AOM_Default instance, as shown in Figure 3-46 .

Figure 3-45 Custom Prometheus common instance

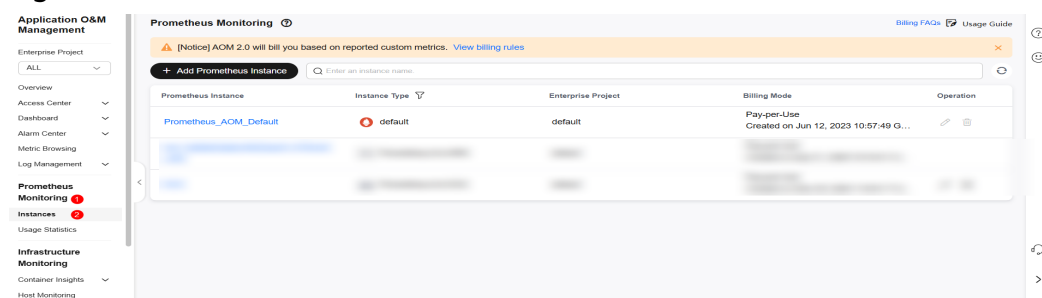
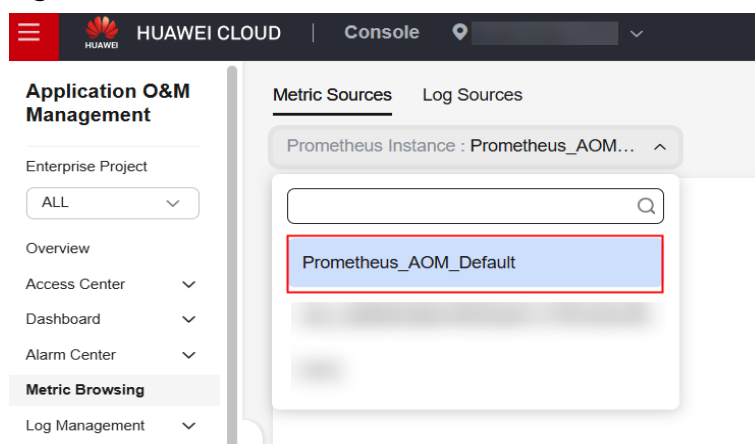


Figure 3-46 Prometheus_AOM_Default instance



3.4.4 AI Suite (NV GPU)

Description

The AI suite, NV GPU, is a device management plug-in that supports GPUs in containers. To use GPU nodes in a cluster, this plug-in must be installed.

Constraints

- When you [create a dedicated resource pool](#), this plug-in is automatically installed only when the instance specification type is set to GPU.
- Do not upgrade the GPU driver before the plug-in upgrade is complete. Otherwise, the driver upgrade may be suspended or fail.

Verifying the Add-on

After the add-on is installed, run the **nvidia-smi** command on the GPU node and the container that schedules GPU resources to verify the availability of the GPU device and driver.

- GPU node:
 - If the add-on version is earlier than 2.0.0, run the following command:
`cd /opt/cloud/cce/nvidia/bin && ./nvidia-smi`
 - If the add-on version is 2.0.0 or later, run the following command:
`cd /usr/local/nvidia/bin && ./nvidia-smi`
- Container:
 - If the cluster version is v1.27 or earlier, run the following command:
`cd /usr/local/nvidia/bin && ./nvidia-smi`
 - If the cluster version is v1.28 or later, run the following command:
`cd /usr/bin && ./nvidia-smi`

If GPU information is returned, the device is available and the add-on has been installed.

```
+-----+
| NVIDIA-SMI 440.118.02    Driver Version: 440.118.02    CUDA Version: 10.2    |
+-----+-----+
| GPU  Name          Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf    Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
+-----+-----+
|  0 Tesla V100-SXM2...    Off   | 00000000:21:01.0 Off  |      0          0 |
| N/A   31C    P0      23W / 300W |  0MiB / 16160MiB |      0%      Default |
+-----+-----+

+-----+
| Processes:                      GPU Memory |
| GPU       PID    Type    Process name      Usage    |
+-----+-----+
| No running processes found      |
+-----+
```

Components

Table 3-22 Plug-in components

Component	Description	Resource Type
nvidia-driver-installer	A workload for installing the NV GPU driver on a node, which only uses resources during the installation process. Once the installation is finished, no resources are used.	Daemon Set

Component	Description	Resource Type
hce20-nvidia-driver-installer	A workload for installing the NV GPU driver on a node, which only uses resources during the installation process. Once the installation is finished, no resources are used (used to adapt to OS HCE 2.0).	Daemon Set
ubuntu22-nvidia-driver-installer	A workload for installing the NV GPU driver on a node, which only uses resources during the installation process. Once the installation is finished, no resources are used (used to adapt to OS Ubuntu22).	Daemon Set
nvidia-gpu-device-plugin	A Kubernetes device plugin that provides NV GPU heterogeneous compute for containers	Daemon Set
nvidia-operator	A component that provides NV GPU node management capabilities for clusters	Deployment
dcgm-exporter	A component that is installed when DCGM-Exporter is enabled to observe DCGM metrics. It is used to collect GPU metrics.	Daemon Set

Change History

Table 3-23 Release history

Plug-in Version	New Feature
2.7.63	Fixed security vulnerabilities.
2.7.42	Added the NV 535.216.03 driver to support xGPUs.
2.6.4	Updated the isolation logic of GPUs.
2.0.72	Updated the isolation logic of GPUs.
2.0.48	Fixed the issue occurred during driver installation.
2.0.44	- Supported NV driver 535. - Non-root users can use xGPUs. - Optimized startup logic.
2.0.14	- Supported xGPU device monitoring. - Supported the compatibility between nvidia.com/gpu and volcano.sh/gpu-* api.

Plug-in Version	New Feature
1.2.29	- Adapted to Ubuntu 22.04. - Optimized the automatic mounting of the GPU driver directory.
1.2.24	- Supported GPU driver version configuration for node pools. - Supported GPU metric collection.
1.2.20	Set the plug-in alias to gpu .
1.2.15	Adapted to CCE v1.23 clusters.

3.4.5 AI Suite (Modelarts Device Plugin)

Description

The AI suite, Modelarts Device Plugin, is a device management plug-in that supports Huawei NPUs in containers.

Constraints

- When you [create a dedicated resource pool](#), this plug-in is automatically installed only when the instance specification type is set to Ascend.
- Job scheduling will be affected if you upgrade this plug-in when creating a notebook instance, creating a training job, deploying a real-time service, or deploying a batch service.

Components

Table 3-24 Plug-in components

Component	Description	Resource Type
modelarts-device-plugin	Allows containers to use Modelarts Device Plugin devices.	Daemon Set

Change History

Table 3-25 Release history

Plug-in Version	New Feature
7.2.0	Kubernetes v1.31 is supported. A2 single-PU multi-pods are supported.

Plug-in Version	New Feature
7.0.1	Kubernetes v1.31 is supported.
7.0.0	Kubernetes v1.31 is supported.
2.7.63	Fixed security vulnerabilities.
2.7.42	Kubernetes v1.31 is supported.
2.1.23	Fixed some issues.
2.1.22	- Fixed some page display issues. - Supernode information can be obtained. - NPU topology information can be reported. - Fixed log printing issues.
2.1.5	- Adapted to CCE v1.29 clusters. - Added silent fault codes.
1.2.15	v1.23 clusters are supported.
1.2.14	Supported NPU monitoring.
1.2.5	Supported automatic installation of NPU drivers.

3.4.6 Volcano Scheduler

Description

Volcano is a batch scheduling platform based on Kubernetes. It provides a series of features required by machine learning, deep learning, bioinformatics, genomics, and other big data applications, as a powerful supplement to Kubernetes capabilities.

Volcano provides general computing capabilities such as high-performance job scheduling, heterogeneous chip management, and job running management. It accesses the computing frameworks for various industries such as AI, big data, gene, and rendering and schedules up to 1,000 pods per second for end users, greatly improving scheduling efficiency and resource utilization.

Volcano provides job scheduling, job management, and queue management for computing applications. Its main features are as follows:

- Diverse computing frameworks: CRD provides common APIs for batch computing tasks. With various plug-ins and advanced job lifecycle management, computing frameworks such as TensorFlow, MPI, and Spark can run on Kubernetes in containers.
- Advanced scheduling: Advanced scheduling capabilities are provided for batch computing and high-performance computing scenarios, including group scheduling, priority preemption, packing, resource reservation, and task topology.

- Queue management: Queues can be effectively managed for scheduling jobs. Complex job scheduling can be implemented based on queue priorities or through multi-level queues.

Volcano has been open-sourced in GitHub at <https://github.com/volcano-sh/volcano>.

Constraints

When upgrading the plug-in, exercise caution when you downgrade a later version to an earlier version, as this may cause job scheduling failures.

Installing a Plug-in

When you [create a dedicated resource pool](#), this plug-in is automatically installed when **Job Type** is set to **Training Jobs**.

Components

Table 3-26 Plug-in components

Component	Description	Resource Type
volcano-scheduler	Schedule pods.	Deployment
volcano-controller	Synchronize CRDs.	Deployment
volcano-admission	Webhook server, which verifies and modifies resources such as pods and jobs	Deployment

Change History

Table 3-27 Release history

Plug-in Version	New Feature
1.17.11	• Optimized the cabinet affinity and packing capabilities. • Optimized the Ascend NPU preemption capability. • Supported Kubernetes v1.32. • Supported topology affinity scheduling of Ascend high-density models.
1.16.8	• Optimized the resource scheduling capability of supernodes. • Kubernetes v1.31 is supported.

Plug-in Version	New Feature
1.15.8	Supported Ascend NPU dual-die affinity scheduling.
1.15.6	Resources can be oversubscribed based on pod profiling.
1.13.5	- Supported scale-in of customized resources based on node priorities. - Optimized the association between preemption and node scale-out.
1.12.18	- Adapted to CCE 1.29 clusters. - The preemption function is enabled by default.
1.12.1	Optimized application auto scaling performance.
1.11.9	- Optimized sorting capability of NPU rank table. - Supported priority-based scheduling in autoscaling scenarios.
1.10.10	Fixed the issue that the local PV plug-in fails to calculate the number of pods pre-bound to the node.
1.10.7	Fixed the issue that the local PV plug-in fails to calculate the number of pods pre-bound to the node.
1.7.1	Supported clusters v1.25.

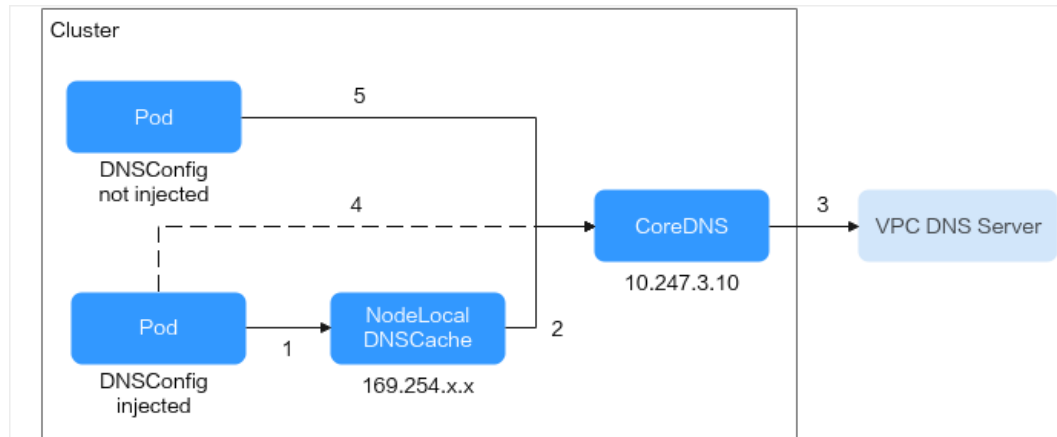
3.4.7 NodeLocal DNSCache

Description

NodeLocal DNSCache is a plug-in developed based on the community NodeLocal DNSCache. It functions as a daemon program that runs the DNS cache proxy on cluster nodes to improve cluster DNS performance.

After NodeLocal DNSCache is enabled, a DNS query goes through the path as shown below.

Figure 3-47 NodeLocal DNSCache query path



The resolution lines are described as follows:

- 1. By default, the pods with DNSConfig injected use NodeLocal DNSCache to resolve requested domain names.
- 2. If NodeLocal DNSCache cannot resolve domain names, it will ask CoreDNS for resolution.
- 3. CoreDNS resolves domain names outside the cluster by using the DNS server in the VPC.
- 4. If a pod with DNSConfig injected cannot access NodeLocal DNSCache, CoreDNS will resolve the domain name.
- 5. By default, CoreDNS resolves domain names for the pods without DNSConfig injected.

Constraints

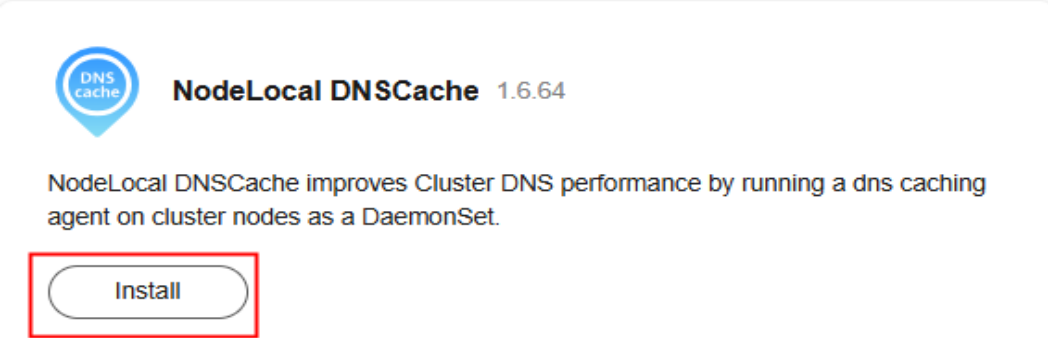
- NodeLocal DNSCache can be installed in ModelArts Standard dedicated resource pool.
- Supported CCE versions: v1.25 to v1.31

Installing a Plug-in

Install the specified plug-in in the resource pool.

1. Log in to the ModelArts console. In the navigation pane on the left, choose **Standard Cluster**.
2. Click the resource pool to access its details page.
3. On the resource pool details page, click the **Plug-ins** tab.
4. Locate the plug-in to be installed in the list and click **Install**.

Figure 3-48 Installing a plug-in



5. In the displayed dialog box, configure the parameters.

Table 3-28 Parameters for configuring NodeLocal DNSCache

Parameter	Sub-Parameter	Description
Specifications	Plug-in Version	Version of NodeLocal DNSCache to be deployed. Version 1.6.64 is supported.
	Plug-in Specifications	Preset: Select HA or Standalone . HA: Dual pods for the automatic DNSConfig injection controller, improving plug-in reliability Standalone: Single pod for the automatic DNSConfig injection controller, consuming fewer resources For details, see Configuration List . Custom: Specify the configuration for deploying NodeLocal DNSCache, including the number of copies, CPU application quota/limitation, and memory application quota/limitation.
	Configuration List	Specify the configuration parameters of the NodeLocal DNSCache plug-in. When you select custom specifications, you can specify the number of copies, CPU application quota/limitation, and memory application quota/limitation.

Parameter	Sub-Parameter	Description
Parameter Configuration	DNSConfig	If this function is enabled, the Admission Webhook intercepts pod creation requests from the target namespace and automatically injects a DNSConfig that uses the DNS cache. DNSConfig will be enabled for all namespaces by default, including existing ones (except for the built-in ones) by automatically adding the node-local-dns-injection=enabled label to them. If the DNS configuration already exists in the pod, the pod's DNS configuration will be preferentially used. If this function is disabled, DNSConfig will not be automatically injected when pods are created in all namespaces. The injected pod DNSConfig is not affected.
	IP Address of DNS Server (Optional)	IP address list of the queried DNS server when the container resolves the domain name. You need to configure this parameter if DNSConfig is enabled. NodeLocal DNSCache and CoreDNS IP addresses are added by default. You can add an additional IP address. Repeated IP addresses will be deleted.
	Search Domain (Optional)	Queried DNS search domain list of the host name in the pod. You need to configure this parameter if DNSConfig is enabled. This parameter is optional. When specified, the provided list will be merged into the base search domain names generated from the chosen DNS policy. Duplicate domain names are removed. Kubernetes allows for at most 3 search domains.
	ndots (Optional)	You need to configure this parameter if DNSConfig is enabled. If a domain name has fewer periods (.) than the specified value of this parameter, the domain name will be combined with the search domain list for DNS query. If it still cannot be resolved, the domain name will be used for DNS query. If the number of periods (.) in a domain name is greater than or equal to the value of ndots, the domain name will be tried first for DNS query. If the domain name cannot be resolved, DNS queries will be attempted by combining the domain name with each domain in the search list.

6. Read "Usage Notes" and select **I have read and understand the preceding information.**
7. Click **OK.**

Components

Table 3-29 Parameters for configuring NodeLocal DNSCache

Component	Description	Resource Type
node-local-dns-admission-controller	Automatic DNSConfig injecting	Deployment
node-local-dns-cache	DNS cache proxy on nodes to improve the DNS performance of the cluster	Daemon Set

Related Operations

For details, see [Viewing the Plug-ins of a Standard Resource Pool on the Resource Pool Details Page](#).

Change History

Table 3-30 Release history

Plug-in Version	New Feature
1.6.66	CCE 1.31 clusters are supported. The admission-controller logs have been updated to standard output logs, and node-local-dns supports custom health check ports.
1.6.64	CCE 1.31 clusters are supported. The admission-controller logs have been updated to standard output logs. node-local-dns supports custom health check ports.
1.6.63	CCE 1.31 clusters are supported. The admission-controller logs have been updated to standard output logs. node-local-dns supports custom health check ports.
1.6.62	CCE 1.31 clusters are supported. The admission-controller logs have been updated to standard output logs, and node-local-dns supports custom health check ports.

3.4.8 Cloud Native Log Collection

Description

The Cloud Native Log Collection plug-in (formerly log-agent) is developed based on Fluent Bit and OpenTelemetry for collecting logs and Kubernetes events. The plug-in can collect standard output logs of training and inference instances in a cluster to LTS.

Log Collection Reliability

The log system's main purpose is to record all stages of data for service components, including startup, initialization, exit, runtime details, and exceptions. It is primarily employed in O&M scenarios for tasks like checking component status and analyzing fault causes.

Standard streams (stdout and stderr) use non-persistent storage. However, data integrity may be compromised due to the following risks:

- Log rotation and compression potentially deleting old files
- Temporary storage volumes being cleared when Kubernetes pods end
- Automatic OS cleanup triggered by limited node storage space

While Cloud Native Log Collection employs techniques like multi-level buffering, priority queues, and resumable uploads to enhance log collection reliability, logs could still be lost in the following situations:

- The service log throughput surpasses the collector's processing capacity.
- The service pod is abruptly terminated and reclaimed by CCE.
- The log collector pod experiences exceptions.

Based on the best practices of cloud native log collection, the following suggestions are provided:

- Use dedicated, high-reliable streams to record critical service data (for example, financial transactions) and store the data in persistent storage.
- Do not store sensitive information such as customer details, payment credentials, and session tokens in logs.

Constraints

You are advised to install 1.7.3 or later.

Supported CCE versions: v1.21 to v1.32

Plug-in Performance Specifications

Performance Item	Description	Remarks
Size of a log	A single log cannot be larger than 512 KB. If multi-line logs are collected, the length of each line will be calculated separately.	N/A
Maximum number of collected files	On a single node, no more than 4,095 files can be listened by all log collection rules.	N/A
Configuration update	Configuration updates take effect in 1 to 3 minutes.	N/A

Installing a Plug-in

Install the specified plug-in in the resource pool.

1. Log in to the ModelArts console. In the navigation pane on the left, choose **Standard Cluster**.
2. Click the resource pool to access its details page.
3. On the resource pool details page, click the **Plug-ins** tab.
4. Locate the plug-in to be installed in the list and click **Install**.
5. In the displayed dialog box, configure the parameters.

Table 3-31 Parameters for configuring Cloud Native Log Collection

Parameter	Sub-Parameter	Description
Specifications	Plug-in Version	Version of Cloud Native Log Collection to be deployed. Version 1.7.3 is supported.
	Plug-in Specifications	Preset: Select Small or Large . Small: A cluster that supports a maximum of 5,000 logs per second Large: A cluster that supports a maximum of 10,000 logs per second Custom: You can adjust the number of plug-in instances and resource quotas as required. High availability is not possible with a single instance. If an error occurs on the node where the plug-in instance runs, the plug-in will fail.
	Configuration List	Detailed configurations of the specified specifications

Parameter	Sub-Parameter	Description
Parameter Configuration	Log Group	Select a log group from the drop-down list. A log group is the basic unit for LTS to manage logs.
	Log Stream	Select a log stream from the drop-down list. A log stream is the basic unit for log reads and writes. If there are many logs to collect, you are advised to separate logs into different log streams based on log types, and name log streams in an easily identifiable way.
	Collect Logical Subpool Logs	Logs for logical subpools are not collected by default. Once this function is enabled, you can collect logs for each logical subpool and set the collection policy. Click Add Logical Pool , select a created logical pool and the corresponding log group and log stream.

6. Read "Usage Notes" and select **I have read and understand the preceding information**.
7. Click **OK**.

Components

Component	Description	Resource Type
fluent-bit	Lightweight log collector and forwarder deployed on each node to collect logs. In 1.5.0 and later versions, logs are directly reported to LTS.	Daemon Set
cop-logs	Used to generate soft links for collected files and run in the same pod as fluent-bit.	Daemon Set
log-operator	Used to generate internal configuration files.	Deployment
otel-collector	Used to collect Kubernetes events and report them to LTS and AOM, and receive and report logs to LTS. The log reporting scope depends on the plug-in version. In 1.5.1 and later versions, this component reports only workload logs that are scaled to CCI.	Deployment

Related Operations

For details, see [Viewing the Plug-ins of a Standard Resource Pool on the Resource Pool Details Page](#).

Change History

Plug-in Version	Supported CCE Cluster Versions	New Feature
1.7.3	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29 v1.30 v1.31	Collecting standard output logs of containers is supported.
1.7.2	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29 v1.30 v1.31	Logs can be compressed in gzip format and sent to LTS.

3.4.9 kube-prometheus-stack

Introduction

The Cloud Native Cluster Monitoring add-on (formerly kube-prometheus-stack) uses Prometheus-operator and Prometheus and provides easy-to-use, end-to-end Kubernetes cluster monitoring.

This add-on enables monitoring data to connect to the monitoring center, so that you can view monitoring data and configure alarms on the console.

Open-source community: <https://github.com/prometheus/prometheus>

Scenario

- If you need to report resource pool metrics to a third-party platform, you can install this plug-in. If this plug-in is not installed, metrics will be reported to Huawei Cloud AOM.

If neither of the preceding scenarios applies, you do not need to install the plug-in.

Constraints

The kube-state-metrics component of the plug-in does not collect labels or annotations of Kubernetes resources.

Permissions

The node-exporter component of this plug-in needs to read the Docker info data from the `/var/run/docker.sock` directory on the host for monitoring the Docker disk space.

The `cap_dac_override` privilege is required for running node-exporter to read Docker info data.

Installing a Plug-in

1. Log in to the [ModelArts console](#). In the navigation pane on the left, choose **Standard Cluster** under **Resource Management**.
2. Click the resource pool name to access its details page.
3. On the resource pool details page, click the **Plug-ins** tab.
4. Locate the plug-in to be installed in the list and click **Install**.

Figure 3-49 Installing a plug-in

Uninstalled plug-ins

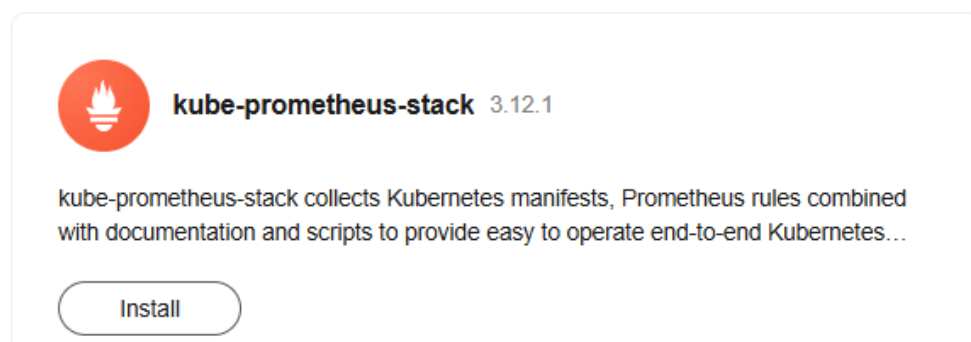


Table 3-32 Parameters

Parameter	Sub-Parameter	Description
Data Storage Configuration (Select at Least One Item)	Report Monitoring Data to a Third-Party Platform	To report Prometheus data to a third-party monitoring system, enter the address and token of the third-party monitoring system and determine whether to skip certificate authentication. • Data Reporting Address: Enter a complete RemoteWrite address of a third-party platform or Huawei Cloud Prometheus instance. • Authentication Mode: Select the authentication mode for reporting monitoring data and enter the username and password of the third-party platform in each authentication mode.
	Local data storage	Local data storage: Monitoring data is stored within the cluster for local metric-based queries. This will use significant CPU, memory, and disk resources. The amount is directly proportional to the size of the cluster and the number of custom metrics being used. No local data storage: Monitoring data is stored outside of the cluster, in either AOM or a third-party monitoring system, to save cluster resources. If this function is enabled, monitoring data is stored in attached EVS or DSS disks so that functions relying on local monitoring data (for example, HPA using custom metrics) can run properly. If you uninstall the plug-in, data on the EVS disk is automatically deleted. To use HPA function, enable this option. Created storage volumes will be billed and counted towards the storage quota. Once enabled, you need to set the EVS disk type and capacity. Local data storage: Monitoring data is stored within the cluster for local metric-based queries. This will use significant CPU, memory, and disk resources. The amount is directly proportional to the size of the cluster and the number of custom metrics being used. No local data storage: Monitoring data is stored outside of the cluster, in either AOM or a third-party monitoring system, to save cluster resources.

Parameter	Sub-Parameter	Description
Specifications	Plug-in Version	Specify the version of the plug-in to be deployed.
	Plug-in Specifications	Preset: • Demo: applicable to clusters with less than 100 Pods • Small: applicable to clusters with less than 2,000 Pods • Medium: applicable to clusters with less than 5,000 Pods • Large: applicable to clusters with more than 5,000 Pods. Custom: Set the CPU and memory quotas as required. Ensure the cluster has sufficient node resources. Otherwise, plug-in instances will fail to be scheduled.
	High Availability	Select Local data storage. Deploy prometheusOperator, prometheus-server, alertmanager, thanosSidecar, thanosQuery, adapter, and kubeStateMetrics in multi-instance mode. Alternatively, do not select Local data storage, and deploy prometheusOperator, prometheus-lightweight, and kubeStateMetrics in multi-instance mode. The supported deployment modes vary depending on the component. For details, see Components.
	Configuration List	Detailed configurations of the specified specifications.
Parameter Settings	Collection Period (s)	If the cluster scale is large (≥ 200 nodes or $\geq 10,000$ Pods), set this parameter to 60 or 30 .
	Data Retention Period	Set the data retention period.

Parameter	Sub-Parameter	Description
	Advanced Settings	• Node-Exporter Listening Port: This port uses the host network and allows Prometheus to collect metrics from the node where the port is located. If the port conflicts with another application port, you can modify it. • Scheduling Policy: The component can run on the node that tolerates the configured taint. After Prometheus is upgraded, you can configure node affinity and taint tolerance for the Prometheus plug-in group on this page. Currently, only taint key-level tolerance is supported. This allows components to run on nodes with a matching taint key. You can add multiple scheduling policies. Policies with specific component names take priority over the global policy. By default, a scheduling policy is disabled if its affinity node key or toleration node taint key is not configured.

5. Read "Usage Notes" and select **I have read and understand the preceding information**.
6. Click **OK**.

Components

Component	Description	Deployment Mode	Resource Type
prometheusOperator (workload name: prometheus-operator)	Deploys and manages the Prometheus Server based on custom resource definitions (CRDs), and monitors and processes the events related to these CRDs. It is the control center of the entire system.	All	Deployment
prometheus (workload name: prometheus-server)	A Prometheus Server cluster deployed by the operator based on the Prometheus CRDs that can be regarded as StatefulSets.	All	Stateful Set

Component	Description	Deployment Mode	Resource Type
alertmanager (workload name: alertmanager-alertmanager)	Alarm center of the plug-in. It receives alarms sent by Prometheus and manages alarm information by deduplicating, grouping, and distributing.	Local data storage enabled	Stateful Set
thanosSidecar	Available only in HA mode. Runs with prometheus-server in the same pod to implement persistent storage of Prometheus metric data.	Local data storage enabled	Container
thanosQuery	Entry for PromQL query when Prometheus is in HA scenarios. It can delete duplicate metrics from Store or Prometheus.	Local data storage enabled	Deployment
adapter (workload name: custom-metrics-apiserver)	Aggregates custom metrics to the native Kubernetes API server.	Local data storage enabled	Deployment
kubeStateMetrics (workload name: kube-state-metrics)	Converts the Prometheus metric data into a format that can be identified by Kubernetes APIs. By default, the kube-state-metrics component does not collect all labels or annotations of Kubernetes resources. To collect such information, see Collecting All Labels and Annotations of a Pod . If the components run in multiple pods, only one pod provides metrics.	All	Deployment
nodeExporter (workload name: node-exporter)	Deployed on each node to collect node monitoring data.	All	DaemonSet

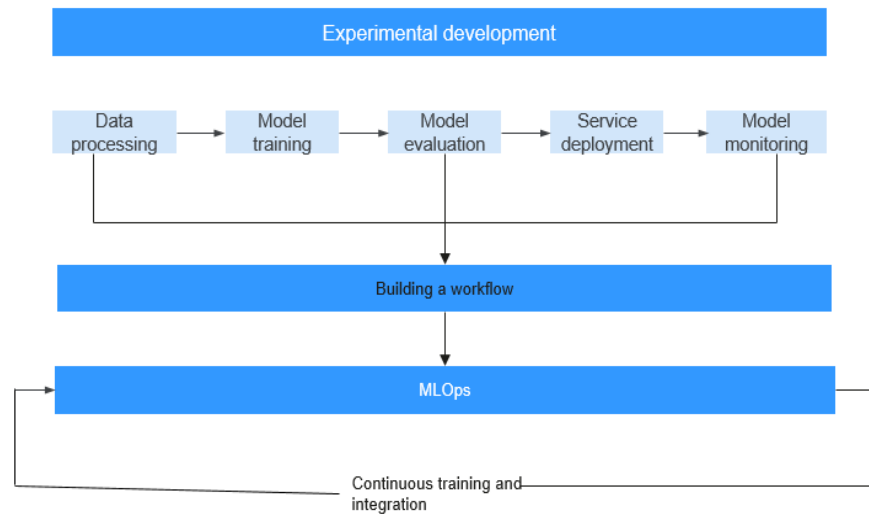
4 Using Workflows for Low-Code AI Development

4.1 What Is Workflow?

MLOps Overview

Understanding MLOps is essential before learning about workflows.

Machine Learning Operations (MLOps) are a set of practices with machine learning (ML) and DevOps combined. The ML development process consists of project design, data engineering, model building, and model deployment. AI development is not a unidirectional pipeline job. During development, multiple iterations of experiments are performed based on the data and model results. To achieve better model results, algorithm engineers perform diverse data processing and model optimization based on the data features and labels of existing datasets. Traditional AI development ends with a one-off delivery of the final model output by iterative experimentation. As time passes after an application is released however, model drift occurs, leading to worsening effects when applying new data and features to the existing model. Iterative experimentation of MLOps forms a fixed pipeline which contains data engineering, model algorithms, and training configurations. You can use the pipeline to continuously perform iterative training on data that is being continuously generated. This ensures that the model, built using the pipeline, is always in an optimum state.

Figure 4-1 MLOps

An entire MLOps link, which covers everything from algorithm development to service delivery and O&M, requires an implementation tool. Originally, the development and delivery processes were conducted separately. The models developed by algorithm engineers were delivered to downstream system engineers. In this process, algorithm engineers are highly involved, which is different from MLOps. There are general delivery cooperation rules in each enterprise. When it comes to project management, working process management needs to be added to AI projects as the system does not simply build and manage pipelines, but acts as a job management system.

The tool for the MLOps link must support the following features:

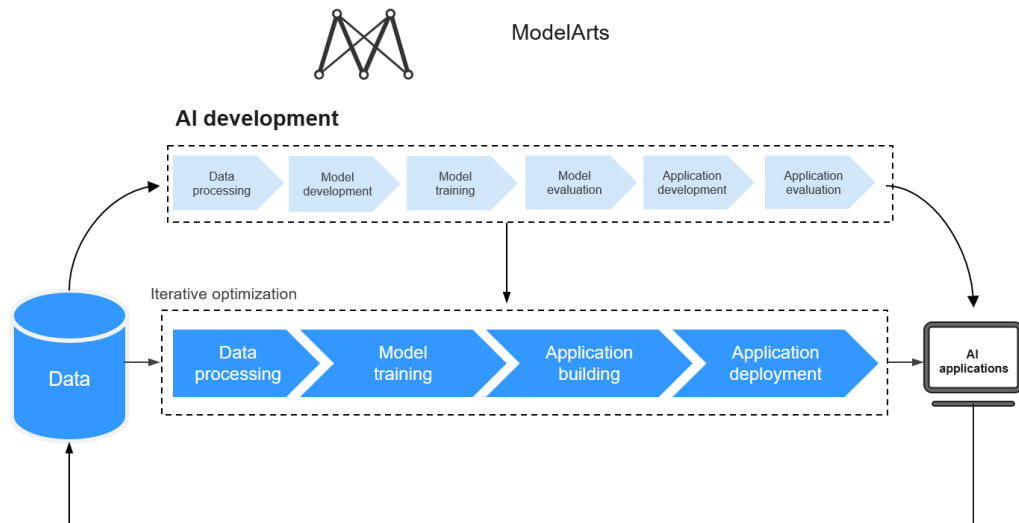
- Process analysis: Accumulated industry sample pipelines help you quickly design AI projects and processes.
- Process definition and redefinition: You can use pipelines to quickly define AI projects and design workflows for model training and release for inference.
- Resource allocation: You can use account management to allocate resource quotas and permissions to participants (including developers and O&M personnel) in the pipeline and view resource usage.
- Task arrangement: Sub-tasks can be arranged based on sub-pipelines. Additionally, notifications can be enabled for efficient management and collaboration.
- Process quality and efficiency evaluation: Pipeline execution views are provided, and checkpoints for different phases such as data evaluation, model evaluation, and performance evaluation are added so that AI project managers can easily view the quality and efficiency of the pipeline execution.
- Process optimization: In each iteration of the pipeline, you can customize core metrics and obtain affected data and causes. In this way, you can quickly determine the next iteration based on these metrics.

Workflow Overview

A workflow is a pipeline tool developed based on service scenarios for deploying models or applications. In ML, a pipeline may involve data labeling, data

processing, model development and training, model evaluation, application development, and application evaluation.

Figure 4-2 Workflow



Different from traditional ML-based model building, workflows can be used to develop production pipelines. Based on MLOps, workflows enable runtime recording, monitoring, and continuous running. The development and continuous iteration of a workflow are separated in products based on roles and concepts.

A pipeline consists of multiple phases. The functions required by the pipeline and the function parameters are called through workflow SDKs. When developing a pipeline, you can use SDKs to describe phases and the relationships between phases. Developing a pipeline is the development state of the workflow. After a pipeline is determined, you can consolidate and provide it for others to use. You do not need to pay attention to what algorithms are used in the pipeline or how the pipeline is implemented. Instead, you only need to check whether the models or applications produced by the pipeline meet the release requirements. If not, you need to check whether the data and parameters need to be adjusted for iteration. Using such a consolidated pipeline is the running state of the workflow.

The development and running states of a workflow are as follows:

- Development state: Workflow Python SDKs are used to develop and debug a pipeline.
- Running state: You can configure and run a produced pipeline in visualized mode.

Leveraging DevOps principles and practices, workflows orchestrate ModelArts capabilities to help you efficiently train, develop, and deploy AI models.

Different functions are implemented in the development and running states of a workflow.

Workflow Development State

Based on service requirements, you can use Python SDKs provided by ModelArts workflows to offer each ModelArts capability as a step in a pipeline. This is a familiar and flexible development mode for AI developers. Python SDKs support:

- Development and building: You can use Python code to create and orchestrate workflows with flexibility.
- Commissioning: The debug and run modes are supported. The run mode supports partial execution and fully execution of a workflow.
- Publishing: The debugged workflows can be fixed and published to the running state for configuration and execution.
- Experiment record: for persistence and the management of experiments.
- Sharing: Workflows can be published to AI Gallery as assets and shared with other users.

Workflow Running State

Workflows are executed in visualized mode. You only need to pay attention to some simple parameter settings to start a workflow. Running workflows are released from the development state or subscribed to from AI Gallery.

Running workflows are released from the development state or subscribed to from AI Gallery.

A running workflow supports:

- Unified configuration management: The parameters and resources required for a workflow are centrally managed.
- Easy-to-use operations: You can start, stop, retry, copy, and delete workflows.
- Running record: records historical running parameters and statuses of the workflow.

Workflow Components

A workflow is the description of a directed acyclic graph (DAG). You can develop a DAG through a workflow. A DAG consists of phases and the relationships between phases. To define a DAG, specify the execution content and sequence on phases. A green rectangle indicates a phase, and the link between phases shows the phase relationship. A DAG is actually an ordered job execution template.

Sample Workflows

ModelArts provides abundant scenario-oriented sample workflows. You can subscribe to them in [AI Gallery](#).

4.2 Managing a Workflow

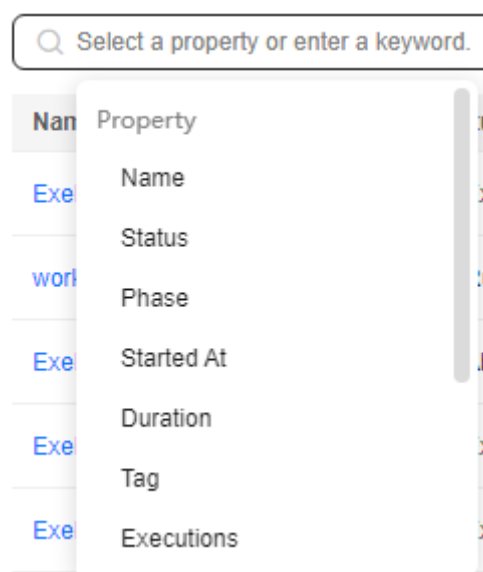
4.2.1 Searching for a Workflow

Procedure

On the workflow list page, you can use the search box to quickly search for workflows based on workflow properties.

1. Log in to the ModelArts console. In the navigation pane, choose **Development Workspace > Workflow**.
2. In the search box above the workflow list, filter workflows based on the required property, such as the name, status, current phase, start time, running duration, or tag.

Figure 4-3 Property



3. Click  on the right of the search box to set the content you want to display on the workflow list page and modify other display settings.
 - **Table Text Wrapping:** This feature is disabled by default. If you enable this feature, excess text will move down to the next line; otherwise, the text will be truncated.
 - **Operation Column:** This feature is enabled by default. If you enable this feature, the **Operation** column is always fixed at the rightmost position of the table.
 - **Custom Columns:** By default, all items are selected. You can select columns you want to see.
4. Click **OK**.
5. To arrange workflows by a specific property, click  in the table header.

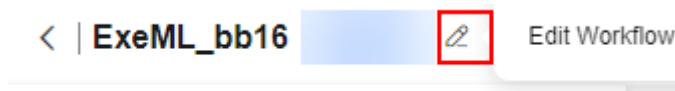
Editing a Workflow Name and Tag

You can rename a workflow and add a tag to make it easier to find.

1. On the ModelArts console, choose **Development Workspace > Workflow** from the navigation pane. The workflow list page is displayed.

2. On the workflow list page, click the name of the target workflow.
3. Click  in the upper left corner.

Figure 4-4 Editing a workflow



4. In the displayed dialog box, modify the workflow name and tag.
Enter a tag and click **Add Tag**. The new tag is displayed below. You can add multiple tags at a time. After the tags are added, click **Yes**.

Figure 4-5 Adding a tag

Edit Workflow

* Workflow Name

Description 0/500

Tag

ExeML

× Add Tag

ExeML

No
Yes

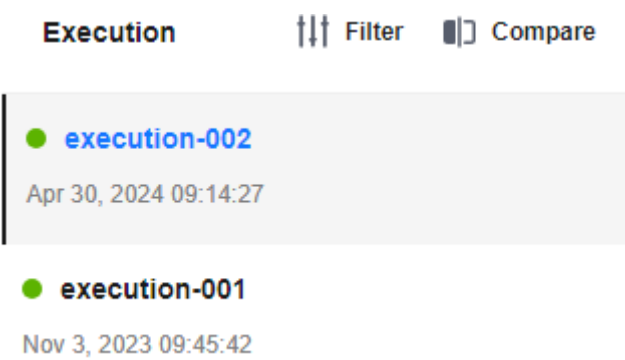
5. Workflows with tags can be filtered by tag in the search box.

4.2.2 Viewing the Running Records of a Workflow

All runtime statuses of a workflow are recorded.

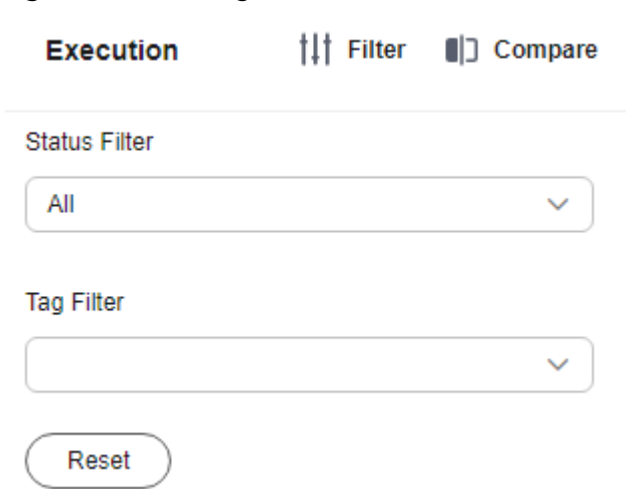
1. On the workflow list page, click the name of the target workflow.
2. On the workflow details page, view all runtime records of the workflow in the left pane.

Figure 4-6 Viewing execution records



- 3. Delete or edit the runtime records, or rerun the workflow.
 - To delete a runtime record that is no longer needed, click **Delete**. In the displayed dialog box, click **Yes**.
 - To distinguish a runtime record from others, click **Edit Tag** to add a tag to it.
 - To rerun the workflow, click **Rerun** on a runtime record.
- 4. Filter and compare all runtime records of the workflow.
 - **Filter**: You can filter all runtime records by status or tag.

Figure 4-7 Filtering



- **Compare**: You can compare all runtime records by status, execution record, start time, duration, and metrics.

Figure 4-8 Comparison

Execution Comparison

Only Show Selected ☐

☐	Name	Status	Execution	Phase	Started At	Duration	recall	top-1	top-5	Metrics	accuracy	f1_score	precision
☐	execution-002	Executed	execution-002		Apr 30, 2024 09:18:08	00:06:35	0.989836363636	0.9898333333	1		0.9898333333	0.987103174803	0.98984219263
☐	execution-002	Executed	execution-002		Apr 30, 2024 09:18:08	00:03:00	—	—	—		—	—	—
☐	execution-001	Executed	execution-001		Nov 3, 2023 09:48:29	00:04:30	0.99182307852	0.994791688886	1		0.994791688886	0.994888440139	0.994791688886
☐	execution-001	Executed	execution-001		Nov 3, 2023 09:47:00	00:01:20	—	—	—		—	—	—

After you click **Start** to run a workflow, the execution record list is refreshed. In addition, the data is updated on both the DAG and dashboard. An execution record is added after each startup.

On the workflow details page, you can click any phase to view its information, including attributes, input, output, and parameters.

4.2.3 Managing a Workflow

Starting a Workflow

Log in to the ModelArts console. In the navigation pane, choose **Development Workspace > Workflow**.

You can run a workflow in any of the following ways:

- On the workflow list page, click **Start** in the **Operation** column. In the displayed dialog box, click **OK**.
- On the runtime configuration page, click **Start** in the upper right corner. In the displayed dialog box, click **OK**.
- On the workflow configuration page, click **Start** in the upper right corner. In the displayed dialog box, click **OK**.

NOTE

After a workflow is started, fees will be generated. Pay attention to the instance status. After a workflow is complete, stop it in a timely manner to avoid unnecessary fees.

Stopping a Workflow

Log in to the ModelArts console. In the navigation pane, choose **Development Workspace > Workflow**.

You can stop a running workflow in either of the following ways:

- Workflow list page
When a workflow is running, the **Stop** button is available in the **Operation** column. Click **Stop**. In the displayed dialog box, click **OK**.
- Click the name of a running workflow and click **Stop** in the upper right corner of the displayed page. In the displayed dialog box, click **OK**.

NOTE

- The **Stop** button is available only for a workflow that is running.
- After a workflow is stopped, the associated training jobs and real-time services are also stopped.

Copying a Workflow

A workflow can have only one running instance. If you want to concurrently run a workflow, copy the workflow. To do so, click **More** in the **Operation** column and select **Copy**. In the displayed dialog box, a new name is automatically generated in the format of "Original workflow name_copy".

You can rename the new workflow. Ensure that the name complies with naming specifications.

 NOTE

A workflow name is 1 to 64 characters long, starting with a letter and containing only letters, digits, underscores (_), and hyphens (-).

Deleting a Workflow

Log in to the ModelArts console. In the navigation pane, choose **Development Workspace > Workflow**.

You can delete a workflow in either of the following ways:

- Workflow list page
 - a. Click **More** in the **Operation** column and select **Delete**.
 - b. In the **Delete Workflow** dialog box, confirm the information, enter **DELETE**, and click **OK**.
- Runtime configuration page
 - a. Click **Delete Workflow** in the upper right corner of the page.
 - b. In the **Delete Workflow** dialog box, confirm the information, enter **DELETE**, and click **OK**.

 NOTE

- Deleted workflows cannot be recovered.
- The instances and generated files related to the deletion operation will not be deleted. The running workflow will stop the running instances.
- When a workflow is deleted, its associated real-time services and training jobs are not deleted. You need to manually delete them on the **Model Training > Training Jobs** page and the **Model Deployment > Real-Time Services** page.

4.2.4 Retrying, Stopping, or Running a Workflow Phase

Retrying, Stopping, or Proceeding a Workflow Phase

- Retrying a phase

If executing a single phase failed, you can click **Retry** to re-execute the current phase without restarting the workflow. Before the retry, you can modify configurations on the **Permission Management** page. The modification takes effect after the affected phase is retried.
- Stopping a phase

Click a phase to view its details. On this page, you can stop the running phase.
- Proceeding a phase

If parameters need to be configured during the runtime of a single phase, the phase is awaiting operation. After the parameters are configured, you can click **Proceed** to proceed to the execution of the current phase.

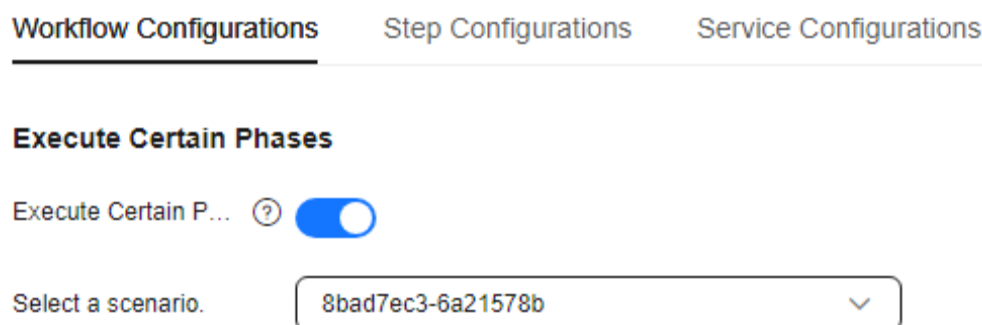
Running Specific Workflow Phases

To reduce the time consumed by repeated execution in large-scale and complex workflows, you can choose specific phases to execute in sequence.

To run specific workflow phases, specify the target phases when developing a workflow. The procedure is as follows:

1. Predefine the phases you want to execute when you use the SDK to create a workflow. For details, see [Specifying Certain Phases to Run in a Workflow](#).
2. When configuring a workflow, enable **Execute Certain Phases**, select phases to be executed, and configure parameters for these phases.

Figure 4-9 Partial execution



3. After saving the configuration, click **Start** to execute certain phases.

4.3 Workflow Development Command Reference

4.3.1 Core Concepts of Workflow Development

Workflow

A workflow is a DAG that consists of phases and the relationships between phases.

A directed line segment shows the dependency between phases. The dependency decides the order of phase execution. In this example, the workflow runs from left to right after it starts. The DAG can handle the multi-branch structure as well. You can design the DAG flexibly according to the real situation. In the multi-branch situation, phases in parallel branches can run at the same time. For details, see [Configuring Multi-Branch Phase Data](#).

Table 4-1 Workflow

Parameter	Description	Mandatory	Data Type
name	Workflow name. The name can contain a maximum of 64 characters, including only letters, digits, underscores (_), and hyphens (-), and must start with a letter.	Yes	str

Parameter	Description	Mandatory	Data Type
desc	Workflow description	Yes	str
steps	Phases contained in a workflow	Yes	list[Step]
storages	Unified storage objects	No	Storage or list[Storage]
policy	Workflow configuration policy, which is used for partial execution	No	Policy

Step

A step is the smallest unit of a workflow. In a DAG, a step is also a phase. Different types of steps have different service abilities. The main parts of a step are as follows.

Table 4-2 Step

Parameter	Description	Mandatory	Data Type
name	Phase name. The name can contain a maximum of 64 characters, including only letters, digits, underscores (_), and hyphens (-), and must start with a letter.	Yes	str
title	Title of a phase, which is displayed in the DAG. If this parameter is not configured, the name is displayed by default.	No	str
step_type	Type of a phase, which determines the function of the phase	Yes	enum
inputs	Inputs of a phase	No	AbstractInput or list[AbstractInput]
outputs	Outputs of a phase	No	AbstractOutput or list[AbstractOutput]
properties	Node properties	No	dict
policy	Phase execution policy, which includes the phase scheduling interval, the phase execution timeout interval, and the option to skip phase execution	No	StepPolicy

Parameter	Description	Mandatory	Data Type
depend_steps	List of dependency phases. This parameter determines the DAG structure and phase execution sequence.	No	Step or list[Step]

Table 4-3 StepPolicy

Parameter	Description	Mandatory	Data Type
poll_interval_seconds	Phase scheduling interval. The default value is 1 second.	Yes	str
max_execution_minutes	Phase execution timeout interval. The default value is 10080 minutes, that is, 7 days.	Yes	str
skip_conditions	Conditions that determine whether a phase is skipped	No	Condition or condition list

Step is a superclass of a phase. It has a conceptual role and is not used directly by you. Different types of phase are created based on functions, including **CreateDatasetStep**, **LabelingStep**, **DatasetImportStep**, **ReleaseDatasetStep**, **JobStep**, **ModelStep**, **ServiceStep** and **ConditionStep**. For details, see [Creating Workflow Phases](#).

Data

Data objects are used for phase input and are classified into the following types:

- Actual data objects, which are specified when you create a workflow
 - Dataset: defines existing datasets. This object is used for data labeling and model training.
 - LabelTask: defines existing labeling jobs. This object is usually used for data labeling and dataset version release.
 - OBSPath: defines an OBS path. This object is used for model training, dataset import, and model import.
 - ServiceData: defines an existing service. This object is used only for service update.
 - SWRImage: defines an existing SWR path. This object is used for model registration.
 - GalleryModel: defines a model subscribed from AI Gallery. This object is used for model registration.
- Placeholder data objects, which are specified when a workflow is running

- DatasetPlaceholder: defines datasets to be specified when a workflow is running. This object is used for data labeling and model training.
 - LabelTaskPlaceholder: defines labeling jobs to be specified when a workflow is running. This object is used for data labeling and dataset version release.
 - OBSPlaceholder: defines an OBS path to be specified when a workflow is running. This object is used for model training, dataset import, and model import.
 - ServiceUpdatePlaceholder: defines existing services to be specified when a workflow is running. This object is used only for service update.
 - SWRImagePlaceholder: defines an SWR path to be specified when a workflow is running. This object is used for model registration.
 - ServiceInputPlaceholder: defines model information required for service deployment when a workflow is running. This object is used only for service deployment and update.
 - DataSelector: supports multiple data types. Currently, this object can be used only on the job phase (only OBS or datasets are supported).
- Data selection object:
DataConsumptionSelector: selects a valid output from the outputs of multiple dependency phases as the data input. This object is usually used for conditional branching. (When creating a workflow, the output of which dependency phase will be used as the data input source is not specified. The data input source should be automatically selected based on the actual execution status of the dependency phases.)

Table 4-4 Dataset

Parameter	Description	Mandatory	Data Type
dataset_name	Dataset name	Yes	str
version_name	Dataset version	No	str

Example:

```
example = Dataset(dataset_name = "", version_name = "")  
# Obtain the dataset name and version name from ModelArts datasets.
```

NOTE

When a dataset is used as the input of a phase, configure **version_name** based on service requirements. For example, **version_name** is not required for LabelingStep and ReleaseDatasetStep, but mandatory for JobStep.

Table 4-5 LabelTask

Parameter	Description	Mandatory	Data Type
dataset_name	Dataset name	Yes	str

Parameter	Description	Mandatory	Data Type
task_name	Labeling job name	Yes	str

Example:

```
example = LabelTask(dataset_name = "***", task_name = "***)  
# Obtain the dataset name and labeling job name from ModelArts datasets of the new version.
```

Table 4-6 OBSPath

Parameter	Description	Mandatory	Data Type
obs_path	OBS path	Yes	str, Storage

Example:

```
example = OBSPath(obs_path = "***)  
# Obtain the OBS path from Object Storage Service.
```

Table 4-7 ServiceData

Parameter	Description	Mandatory	Data Type
service_id	Service ID	Yes	str

Example:

```
example = ServiceData(service_id = "***)  
# Obtain the service ID in ModelArts Real-Time Services. This object describes a specified real-time service  
and is used for service update.
```

Table 4-8 SWRImage

Parameter	Description	Mandatory	Data Type
swr_path	SWR path to a container image	Yes	str

Example:

```
example = SWRImage(swr_path = "***)  
# Container image path, which is used as the input for model registration
```

Table 4-9 GalleryModel

Parameter	Description	Mandatory	Data Type
subscription_id	Subscription ID of a subscribed model	Yes	str
version_num	Version number of a subscribed model	Yes	str

Example:

```
example = GalleryModel(subscription_id="***", version_num="**")
# Subscribed model object, which is used as the input of the model registration phase
```

Table 4-10 DatasetPlaceholder

Parameter	Description	Mandatory	Data Type
name	Name	Yes	str
data_type	Data Type	No	DataTypeEnum
delay	Whether the data object is configured when the phase is running. The default value is False .	No	bool
default	Default value of a data object	No	Dataset

Example:

```
example = DatasetPlaceholder(name = "***", data_type = DataTypeEnum.IMAGE_CLASSIFICATION)
# Dataset object placeholder. Configure data_type to specify supported data types.
```

Table 4-11 OBSPlaceholder

Parameter	Description	Mandatory	Data Type
name	Name	Yes	str
object_type	OBS object type. Only "file" and "directory" are supported.	Yes	str
delay	Whether the data object is configured when the phase is running. The default value is False .	No	bool

Parameter	Description	Mandatory	Data Type
default	Default value of a data object	No	OBSPath

Example:

```
example = OBSPlaceholder(name = "***", object_type = "directory" )
# OBS object placeholder. You can set object_type to file or directory.
```

Table 4-12 LabelTaskPlaceholder

Parameter	Description	Mandatory	Data Type
name	Name	Yes	str
task_type	Type of a labeling job	No	LabelTaskTypeEnum
delay	Whether the data object is configured when the phase is running. The default value is False .	No	bool

Example:

```
example = LabelTaskPlaceholder(name = "***")
# LabelTask object placeholder
```

Table 4-13 ServiceUpdatePlaceholder

Parameter	Description	Mandatory	Data Type
name	Parameter	Yes	str
delay	Whether the data object is configured when the phase is running. The default value is False .	No	bool

Example:

```
example = ServiceUpdatePlaceholder(name = "***")
# ServiceData object placeholder, which is used as the input for service update
```

Table 4-14 SWRImagePlaceholder

Parameter	Description	Mandatory	Data Type
name	Name	Yes	str

Parameter	Description	Mandatory	Data Type
delay	Whether the data object is configured when the phase is running. The default value is False .	No	bool

Example:

```
example = SWRImagePlaceholder(name = "***" )  
# SWRImage object placeholder, which is used as the input for model registration
```

Table 4-15 ServiceInputPlaceholder

Parameter	Description	Mandatory	Data Type
name	Name	Yes	str
model_name	Model name	Yes	str or Placeholder
model_version	Model version	No	str
envs	Environment variables	No	dict
delay	Whether service deployment information is configured when the phase is running. The default value is True .	No	bool

Example:

```
example = ServiceInputPlaceholder(name = "***" , model_name = "model_name")  
# This object is used as the input for service deployment or service update.
```

Table 4-16 DataSelector

Parameter	Description	Mandatory	Data Type
name	Name	Yes	str
data_type_list	Supported data types. Currently, only obs and dataset are supported.	Yes	list

Parameter	Description	Mandatory	Data Type
delay	Whether the data object is configured when the phase is running. The default value is False .	No	bool

Example:

```
example = DataSelector(name = "***", data_type_list=["obs", "dataset"])  
# This object is used as the input of the job phase.
```

Table 4-17 DataConsumptionSelector

Parameter	Description	Mandatory	Data Type
data_list	Output data objects of a dependency phase	Yes	list

Example:

```
example = DataConsumptionSelector(data_list=[step1.outputs["step1_output_name"].as_input(),  
step2.outputs["step2_output_name"].as_input()])  
# Use the valid output from either step 1 or step 2 as the input. If step 1 is skipped and has no output, use  
the valid output from step 2 as the input. (Make sure that data_list has only one valid output.)
```

4.3.2 Configuring Workflow Parameters

Description

A workflow parameter is a placeholder object that can be configured when the workflow runs. The following data types are supported: int, str, bool, float, Enum, dict, and list. You can display fields (such as algorithm hyperparameters) in a phase as placeholders in a transparent way. You can modify and use the default values that are set for them.

Parameter Overview (Placeholder)

Parameter	Description	Mandatory	Data Type
name	Parameter name, which must be globally unique.	Yes	str

Parameter	Description	Mandatory	Data Type
placeholder_type	Parameter type. The mapping between placeholder types and actual data types: PlaceholderType.INT -> int PlaceholderType.STR -> str PlaceholderType.BOOL -> bool PlaceholderType.FLOAT -> float PlaceholderType.ENUM -> Enum PlaceholderType.JSON -> dict PlaceholderType.LIST -> list - When the type is PlaceholderType.ENUM, the enum_list field cannot be empty. - When the type is PlaceholderType.LIST, the placeholder_format field cannot be empty and can only be set to str, int, float, or bool, indicating the data types in the list.	Yes	PlaceholderType
default	Default parameter value. The data type must be the same as that of placeholder_type .	No	Any
placeholder_format	Supported data formats. Currently, obs , flavor , train_flavor , swr , and pacific are supported.	No	str
delay	Whether parameters are set when the workflow is running. The default value is False , indicating that parameters are set before the workflow runs. If the value is True , parameters are set in an action of the phase where they are needed.	No	bool
description	Parameter description.	No	str
enum_list	List of enumerated values of a parameter. This parameter is mandatory only for parameters of PlaceholderType.ENUM type.	No	list
constraint	Constraints on parameters. This parameter only supports the constraints of training specifications and is not visible to you.	No	dict

Parameter	Description	Mandatory	Data Type
required	Whether the parameter is mandatory. - The default value is True. - This parameter cannot be set to False for Delay. This parameter is optional at the frontend during execution.	No	bool

Examples

- Integer parameter**

```
from modelarts import workflow as wf
wf.Placeholder(name="placeholder_int", placeholder_type=wf.PlaceholderType.INT, default=1,
description="This is an integer parameter.")
```
- String parameter**

```
from modelarts import workflow as wf
wf.Placeholder(name="placeholder_str", placeholder_type=wf.PlaceholderType.STR,
default="default_value", description="This is a string parameter.")
```
- Bool parameter**

```
from modelarts import workflow as wf
wf.Placeholder(name="placeholder_bool", placeholder_type=wf.PlaceholderType.BOOL, default=True,
description="This is a bool parameter.")
```
- Float parameter**

```
from modelarts import workflow as wf
wf.Placeholder(name="placeholder_float", placeholder_type=wf.PlaceholderType.FLOAT, default=0.1,
description="This is a float parameter.")
```
- Enumeration parameter**

```
from modelarts import workflow as wf
wf.Placeholder(name="placeholder_enum", placeholder_type=wf.PlaceholderType.ENUM, default="a",
enum_list=["a", "b"], description="This is an enumeration parameter.")
```
- Dictionary parameter**

```
from modelarts import workflow as wf
wf.Placeholder(name="placeholder_dict", placeholder_type=wf.PlaceholderType.JSON, default={"key":
"value"}, description="This is a dictionary parameter.")
```
- List parameter**

```
from modelarts import workflow as wf
wf.Placeholder(name="placeholder_list", placeholder_type=wf.PlaceholderType.LIST, default=[1, 2],
placeholder_format="int", description="This is a list parameter and its value is an integer.")
```

4.3.3 Configuring the Input and Output Paths of a Workflow

Description

Unified storage is used for workflow directory management. It centrally manages all storage paths of a workflow with these functions:

- Input directory management:** When developing a workflow, you can centrally manage all data storage paths. You can store data and configure the root directory based on your own requirements. This function orchestrates directories but does not create them.
- Output directory management:** When developing a workflow, you can centrally manage all output paths. You do not need to create output

directories. Instead, you only need to configure the root path before the workflow runs and view the output data in the specified directories based on your directory orchestration rules. In addition, multiple executions of the same workflow are output to different directories, isolating data for different executions.

Common Usage

- **InputStorage** (Path concatenation)

This object is used to centrally manage input directories. The following is an example:

```
import modelarts.workflow as wf
storage = wf.data.InputStorage(name="storage_name", title="title_info",
description="description_info") # Only name is mandatory.
input_data = wf.data.OBSPath(obs_path = storage.join("directory_path")) # Add a slash (/) after a
directory, for example, storage.join("/input/data/").
```

When a workflow is running, if the root path of the storage object is **/root/**, the obtained path will be **/root/directory_path**.

- **OutputStorage** (Directory creation)

This object is used to centrally manage output directories and ensure that multiple executions of the same workflow are output to different directories. The following is an example:

```
import modelarts.workflow as wf
storage = wf.data.OutputStorage(name="storage_name", title="title_info",
description="description_info") # Only name is mandatory.
output_path = wf.data.OBSOutputConfig(obs_path = storage.join("directory_path")) # Only a
directory can be created but not files.
```

When a workflow is running, if the root path of the storage object is set to **/root/**, the system will automatically create a relative directory and the obtained path will be **/root/Execution ID/directory_path**.

Advanced Usage

Storage

This object contains capabilities of InputStorage and OutputStorage and can be flexibly used based on your needs.

Parameter	Description	Mandatory	Data Type
name	Name.	Yes	str
title	If this parameter is left blank, the value of name is used by default.	No	str
description	Description.	No	str
create_dir	Whether to create a directory. The default value is False .	No	bool

Parameter	Description	Mandatory	Data Type
with_execution_id	Whether to combine execution_id when a directory is created. The default value is False . This parameter can be set to True only when create_dir is set to True .	No	bool

The following is an example:

- Implementing **InputStorage** capabilities

```
import modelarts.workflow as wf
# Create a Storage object (with_execution_id=False, create_dir=False).
storage = wf.data.Storage(name="storage_name", title="title_info", description="description_info",
with_execution_id=False, create_dir=False)
input_data = wf.data.OBSPPath(obs_path = storage.join("directory_path")) # Add a slash (/) after a
directory, for example, storage.join("/input/data/").
```

When a workflow is running, if the root path of the storage object is **/root/**, the obtained path will be **/root/directory_path**.

- Implementing **OutputStorage** capabilities

```
import modelarts.workflow as wf
# Create a Storage object (with_execution_id=True, create_dir=True).
storage = wf.data.Storage(name="storage_name", title="title_info", description="description_info",
with_execution_id=True, create_dir=True)
output_path = wf.data.OBSOutputConfig(obs_path = storage.join("directory_path")) # Only a
directory can be created.
```

When a workflow is running, if the root path of the storage object is set to **/root/**, the system will automatically create a relative directory and the obtained path will be **/root/Execution ID/directory_path**.

- Implementing different capabilities of a Storage object through the **join** method

```
import modelarts.workflow as wf
# Create a Storage object. Assume that the root directory of the Storage object is /root/.
storage = wf.data.Storage(name="storage_name", title="title_info", description="description_info",
with_execution_id=False, create_dir=False)
input_data1 = wf.data.OBSPPath(obs_path = storage) # The obtained path is /root/.
input_data2 = wf.data.OBSPPath(obs_path = storage.join("directory_path")) # The obtained path is /root/directory_path. Ensure that the path exists.
output_path1 = wf.data.OBSOutputConfig(obs_path = storage.join(directory="directory_path",
with_execution_id=False, create_dir=True)) # The system automatically creates a directory /root/directory_path.
output_path2 = wf.data.OBSOutputConfig(obs_path = storage.join(directory="directory_path",
with_execution_id=True, create_dir=True)) # The system automatically creates a directory /root/Execution ID/directory_path.
```

Chain call is supported for **Storage**.

The following is an example:

```
import modelarts.workflow as wf
# Create a base class Storage object. Assume that the root directory of the Storage object is /root/.
storage = wf.data.Storage(name="storage_name", title="title_info", description="description_info",
with_execution_id=False, create_dir=False)
input_storage = storage.join("directory_path_1") # The obtained path is /root/directory_path_1.
input_storage_next = input_storage.join("directory_path_2") # The obtained path is /root/directory_path_1/directory_path_2.
```

Examples

Unified storage is mainly used in the job phase. The following code uses a workflow that contains only the training phase as an example.

```
from modelarts import workflow as wf

# Create an InputStorage object. Assume that the root directory of the Storage object is /root/input-data/.
input_storage = wf.data.InputStorage(name="input_storage", title="title_info",
description="description_info") # Only name is mandatory.

# Create an OutputStorage object. Assume that the root directory of the Storage object is /root/output/.
output_storage = wf.data.OutputStorage(name="output_storage_name", title="title_info",
description="description_info") # Only name is mandatory.

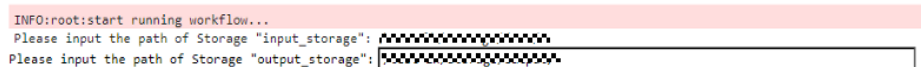
# Use JobStep to define a training phase, and set OBS paths for storing inputs and outputs.
job_step = wf.steps.JobStep(
    name="training_job", # Name of a training phase. The name contains a maximum of 64 characters,
    including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and must be
    unique in a workflow.
    title="Image Classification Training", # Title, which defaults to the value of name
    algorithm=wf.AIGalleryAlgorithm(subscription_id="subscription_ID",
    item_version_id="item_version_ID"), # Algorithm used for training. In this example, an algorithm subscribed
    to from AI Gallery is used.
    inputs=[
        wf.steps.JobInput(name="data_url_1", data=wf.data.OBSPath(obs_path = input_storage.join("/
dataset1/new.manifest"))), # The obtained path is /root/input-data/dataset1/new.manifest.
        wf.steps.JobInput(name="data_url_2", data=wf.data.OBSPath(obs_path = input_storage.join("/
dataset2/new.manifest"))), # The obtained path is /root/input-data/dataset2/new.manifest.
    ],
    outputs=wf.steps.JobOutput(name="train_url",
    obs_config=wf.data.OBSOutputConfig(obs_path=output_storage.join("/model/"))), # The training output
    path is /root/output/Execution ID/model/.
    spec=wf.steps.JobSpec(
        resource=wf.steps.JobResource(
            flavor=wf.Placeholder(name="train_flavor", placeholder_type=wf.PlaceholderType.JSON,
            description="Training flavor")
        ),
        log_export_path=wf.steps.job_step.LogExportPath(obs_url=output_storage.join("/logs/")) # The log
        output path is /root/output/Execution ID/logs/.
    )# Training flavors
)

# Define a workflow that contains only the job phase.
workflow = wf.Workflow(
    name="test-workflow",
    desc="this is a test workflow",
    steps=[job_step],
    storages=[input_storage, output_storage] # Add Storage objects used in this workflow.
)
```

Configuring Root Paths in the Development State

Use the **run** method of the workflow object, and input root paths in the text box that is displayed when the workflow starts to run.

Figure 4-10 Inputting root paths

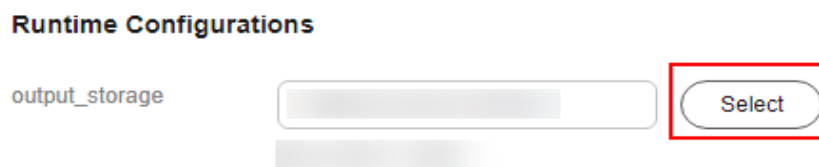


You must enter a valid path. If the path does not exist, an error will occur. The path format must be */Bucket name/Folder path/*.

Configuring Root Paths in the Running State

Use the **release** method of the workflow object to release the workflow to the running state. On the ModelArts console, go to the **Workflow** page, find the target workflow, and configure root paths.

Figure 4-11 Configuring root paths



4.3.4 Creating Workflow Phases

4.3.4.1 Creating a Dataset Phase

Description

This phase integrates capabilities of the ModelArts dataset module, allowing you to create datasets of the new version. This phase is used to centrally manage existing data by creating datasets. It is usually followed by a dataset import phase or a labeling phase.

Parameter Overview

You can use CreateDatasetStep to create a dataset creation phase. The following is an example of defining a CreateDatasetStep.

Table 4-18 CreateDatasetStep

Parameter	Description	Mandatory	Data Type
name	Name of a dataset creation phase. The name contains a maximum of 64 characters, including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and must be unique in a workflow.	Yes	str
inputs	Inputs of the dataset creation phase.	Yes	CreateDatasetInput or a list of CreateDatasetInput

Parameter	Description	Mandatory	Data Type
outputs	Outputs of the dataset creation phase.	Yes	CreateDatasetOutput or a list of CreateDatasetOutput
properties	Configurations for dataset creation.	Yes	DatasetProperties
title	Title for frontend display.	No	str
description	Description of the dataset creation phase.	No	str
policy	Phase execution policy.	No	StepPolicy
depend_steps	Dependent phases.	No	Step or step list

Table 4-19 CreateDatasetInput

Parameter	Description	Mandatory	Data Type
name	Input name of the dataset creation phase. The name can contain a maximum of 64 characters, including only letters, digits, underscores (_), and hyphens (-), and must start with a letter. The input name of a step must be unique.	Yes	str
data	Input data object of the dataset creation phase.	Yes	OBS object. Currently, only OBSPath, OBSConsumption, OBSPlaceholder, and DataConsumption Selector are supported.

Table 4-20 CreateDatasetOutput

Parameter	Description	Mandatory	Data Type
name	Output name of the dataset creation phase. The name can contain a maximum of 64 characters, including only letters, digits, underscores (_), and hyphens (-), and must start with a letter. The output name of a step must be unique.	Yes	str
config	Output configurations of the dataset creation phase.	Yes	Currently, only OBSOutputConfig is supported.

Table 4-21 DatasetProperties

Parameter	Description	Mandatory	Data Type
dataset_name	Dataset name. The value contains 1 to 100 characters. Only letters, digits, underscores (_), and hyphens (-) are allowed.	Yes	str, Placeholder
dataset_format	Dataset format. The default value is 0, indicating the file type.	No	0: file 1: table
data_type	Data type. The default value is FREE_FORMAT .	No	DataTypeEnum
description	Description	No	str
import_data	Whether to import data. The default value is False . Currently, only table data is supported.	No	bool
work_path_type	Type of the dataset output path. Currently, only OBS is supported. The default value is 0 .	No	int
import_config	Configurations for label import. The default value is None . When creating a dataset based on labeled data, you can specify this parameter to import labeling information.	No	ImportConfig

Table 4-22 Importconfig

Parameter	Description	Mandatory	Data Type
import_annotations	Whether to automatically import the labeling information in the input directory, supporting detection, image classification, and text classification. The options are as follows: true: The labeling information in the input directory is imported. (Default) false: The labeling information in the input directory is not imported.	No	str, Placeholder
import_type	Import mode. The options are as follows: dir: imported from an OBS path manifest: imported from a manifest file	No	0: file type ImportTypeEnum
annotation_format_config	Configurations of the imported labeling format.	No	DAnnotationFormatTypeEnumconfig list

Table 4-23 AnnotationFormatConfig

Parameter	Description	Mandatory	Data Type
format_name	Name of a labeling format	No	AnnotationFormatEnum
scene	Labeling scenario, which is optional	No	LabelTaskTypeEnum

Enumeration	Value
ImportTypeEnum	DIR MANIFEST

Enumeration	Value
DataTypeEnum	IMAGE TEXT AUDIO TABULAR VIDEO FREE_FORMAT
AnnotationFormatEnum	MA_IMAGE_CLASSIFICATION_V1 MA_IMAGENET_V1 MA_PASCAL_VOC_V1 YOLO MA_IMAGE_SEGMENTATION_V1 MA_TEXT_CLASSIFICATION_COMBINE_V1 MA_TEXT_CLASSIFICATION_V1 MA_AUDIO_CLASSIFICATION_DIR_V1

Examples

There are two scenarios:

- Creating a dataset using unlabeled data
- Creating a dataset using labeled data with labels imported

Creating a dataset using unlabeled data

Data preparation: Store unlabeled data in an OBS folder.

```
from modelarts import workflow as wf
# Use CreateDatasetStep to create a dataset of the new version using OBS data.

# Define parameters of the dataset output path.
dataset_output_path = wf.Placeholder(name="dataset_output_path",
placeholder_type=wf.PlaceholderType.STR, placeholder_format="obs")

# Define the dataset name.
dataset_name = wf.Placeholder(name="dataset_name", placeholder_type=wf.PlaceholderType.STR)

create_dataset = wf.steps.CreateDatasetStep(
    name="create_dataset", # Name of a dataset creation phase. The name contains a maximum of 64
    characters, including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and
    must be unique in a workflow.
    title="Dataset Creation", # Title, which defaults to the value of name
    inputs=wf.steps.CreateDatasetInput(name="input_name",
    data=wf.data.OBSPlaceholder(name="obs_placeholder_name", object_type="directory")),#
    CreateDatasetStep inputs, configured when the workflow is running; the data field can also be represented
    by the wf.data.OBSPath(obs_path="fake_obs_path") object.
    outputs=wf.steps.CreateDatasetOutput(name="output_name",
    config=wf.data.OBSOutputConfig(obs_path=dataset_output_path)),# CreateDatasetStep outputs
    properties=wf.steps.DatasetProperties(
        dataset_name=dataset_name, # If the dataset name does not exist, a dataset will be created using
        this name. If the dataset name exists, the corresponding dataset will be used.
        data_type=wf.data.DataTypeEnum.IMAGE, # Data type of the dataset, for example, image
```

```
)
)
# Ensure that the dataset name is not used by others under the account. Otherwise, the dataset created by
others will be used in the subsequent phases.

workflow = wf.Workflow(
    name="create-dataset-demo",
    desc="this is a demo workflow",
    steps=[create_dataset]
)
```

Creating a dataset using labeled data with labels imported

Data preparation: Store labeled data in an OBS folder.

For details about specifications for importing labeled data from an OBS directory, see [Specifications for Importing Data from an OBS Directory](#).

```
from modelarts import workflow as wf
# Use CreateDatasetStep to create a dataset of the new version using OBS data.

# Define parameters of the dataset output path.
dataset_output_path = wf.Placeholder(name="dataset_placeholder_name",
placeholder_type=wf.PlaceholderType.STR, placeholder_format="obs")

# Define the dataset name.
dataset_name = wf.Placeholder(name="dataset_placeholder_name",
placeholder_type=wf.PlaceholderType.STR)

create_dataset = wf.steps.CreateDatasetStep(
    name="create_dataset", # Name of a dataset creation phase. The name contains a maximum of 64
characters, including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and
must be unique in a workflow.
    title="Dataset Creation", # Title, which defaults to the value of name
    inputs=wf.steps.CreateDatasetInput(name="input_name",
data=wf.data.OBSPlaceholder(name="obs_placeholder_name", object_type="directory")),#
CreateDatasetStep inputs, configured when the workflow is running; the data field can also be represented
by the wf.data.OBSPath(obs_path="fake_obs_path") object.
    outputs=wf.steps.CreateDatasetOutput(name="output_name",
config=wf.data.OBSOutputConfig(obs_path=dataset_output_path)),# CreateDatasetStep outputs
    properties=wf.steps.DatasetProperties(
        dataset_name=dataset_name, # If the dataset name does not exist, a dataset will be created using
this name. If the dataset name exists, the corresponding dataset will be used.
        data_type=wf.data.DataTypeEnum.IMAGE, # Data type of the dataset, for example, image
        import_config=wf.steps.ImportConfig(
            annotation_format_config=[
                wf.steps.AnnotationFormatConfig(
                    format_name=wf.steps.AnnotationFormatEnum.MA_IMAGE_CLASSIFICATION_V1, # Labeling
format of labeled data
                    scene=wf.data.LabelTaskTypeEnum.IMAGE_CLASSIFICATION # Labeling scene
                )
            ]
        )
    )
)
# Ensure that the dataset name is not used by others under the account. Otherwise, the dataset created by
others will be used in the subsequent phases.

workflow = wf.Workflow(
    name="create-dataset-demo",
    desc="this is a demo workflow",
    steps=[create_dataset]
)
```

4.3.4.2 Creating a Dataset Labeling Phase

Description

This phase integrates capabilities of the ModelArts dataset module, allowing you to label datasets. The labeling phase is used to create labeling jobs or label existing jobs.

Parameter Overview

You can use LabelingStep to create a labeling phase. The following is an example of defining a LabelingStep.

Table 4-24 LabelingStep

Parameter	Description	Mandatory	Data Type
name	Name of a labeling phase. The name contains a maximum of 64 characters, including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and must be unique in a workflow.	Yes	str
inputs	Inputs of the labeling phase.	Yes	LabelingInput or LabelingInput list
outputs	Outputs of the labeling phase.	Yes	LabelingOutput or LabelingOutput list
properties	Configurations for dataset labeling.	Yes	LabelTaskProperties
title	Title for frontend display.	No	str
description	Description of the labeling phase.	No	str
policy	Phase execution policy.	No	StepPolicy
depend_steps	Dependent phases.	No	Step or step list

Table 4-25 LabelingInput

Parameter	Description	Mandatory	Data Type
name	Input name of the labeling phase. The name can contain a maximum of 64 characters, including only letters, digits, underscores (_), and hyphens (-), and must start with a letter. The input name of a step must be unique.	Yes	str
data	Input data object of the labeling phase.	Yes	Dataset or labeling job object. Currently, only Dataset, DatasetConsumption, DatasetPlaceholder, LabelTask, LabelTaskPlaceholder, LabelTaskConsumption, and DataConsumptionSelector are supported.

Table 4-26 LabelingOutput

Parameter	Description	Mandatory	Data Type
name	Output name of the labeling phase. The name can contain a maximum of 64 characters, including only letters, digits, underscores (_), and hyphens (-), and must start with a letter. The output name of a step must be unique.	Yes	str

Table 4-27 LabelTaskProperties

Parameter	Description	Mandatory	Data Type
task_type	Type of a labeling job. Jobs of the specified type are returned.	Yes	LabelTaskTypeEnum
task_name	Labeling job name. The value contains 1 to 100 characters, including only letters, digits, hyphens (-), and underscores (_). This parameter is mandatory when the input is a dataset object.	No	str, Placeholder
labels	Labels to be created.	No	Label
properties	Attributes of a labeling job. You can update this field to record custom information.	No	dict

Parameter	Description	Mandatory	Data Type
auto_sync_dataset	Whether to automatically synchronize the result of a labeling job to the dataset. The options are as follows: • true: The labeling result of the labeling job is automatically synchronized to the dataset. (Default) • false: The labeling result of the labeling job is not automatically synchronized to the dataset.	No	bool
content_labeling	Whether to enable content labeling for speech paragraph labeling. This function is enabled by default.	No	bool
description	Labeling job description. The description contains 0 to 256 characters and does not support the following special characters: ^!<>=&""	No	str

Table 4-28 Label

Parameter	Description	Mandatory	Data Type
name	Tag name	No	str

Parameter	Description	Mandatory	Data Type
property	Basic attribute key-value pair of a label, such as color and shortcut keys	No	str, dic, Placeholder
type	Tag type	No	LabelTypeEnum

Enumeration	Value
LabelTaskTypeEnum	IMAGE_CLASSIFICATION OBJECT_DETECTION IMAGE_SEGMENTATION TEXT_CLASSIFICATION NAMED_ENTITY_RECOGNITION TEXT_TRIPLE AUDIO_CLASSIFICATION SPEECH_CONTENT SPEECH_SEGMENTATION DATASET_TABULAR VIDEO_ANNOTATION FREE_FORMAT

Sample Code of a Dataset Labeling Phase

There are three scenarios:

- Scenario 1: Creating a labeling job for a specified dataset and labeling the dataset

Scenarios:

- You have created only one unlabeled dataset and need to label it when the workflow is running.
- After a dataset is imported, the dataset needs to be labeled.

Data preparation: Create a dataset on the ModelArts console.

```
from modelarts import workflow as wf
```

```
# Use LabelingStep to create a labeling job for the input dataset and label it.
```

```
# Define an input dataset.
```

```
dataset = wf.data.DatasetPlaceholder(name="input_dataset")
```

```
# Define the name parameters of the labeling job.
```

```
task_name = wf.Placeholder(name="placeholder_name", placeholder_type=wf.PlaceholderType.STR)
```

```
labeling = wf.steps.LabelingStep(
```

```
    name="labeling", # Name of the labeling phase. The name contains a maximum of 64 characters, including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and must be
```

```
unique in a workflow.
    title="Dataset Labeling", # Title, which defaults to the value of name
    properties=wf.steps.LabelTaskProperties(
        task_type=wf.data.LabelTaskTypeEnum.IMAGE_CLASSIFICATION, # Labeling job type, for
        example, image classification
        task_name=task_name # If the labeling job name does not exist, a job will be created using
        this name. If the labeling job name exists, the corresponding job will be used.
    ),
    inputs=wf.steps.LabelingInput(name="input_name", data=dataset), # LabelingStep inputs. The
    dataset object is configured when the workflow is running. You can also use
wf.data.Dataset(dataset_name="fake_dataset_name") for the data field.
    outputs=wf.steps.LabelingOutput(name="output_name"), # LabelingStep outputs
)

workflow = wf.Workflow(
    name="labeling-step-demo",
    desc="this is a demo workflow",
    steps=[labeling]
)
```

- Scenario 2: Labeling a specified job

Scenarios:

- You have created a labeling job and need to label it when the workflow is running.
- After a dataset is imported, the dataset needs to be labeled.

Data preparation: Create a labeling job using a specified dataset on the ModelArts console.

```
from modelarts import workflow as wf
# Input a labeling job and label it.
```

```
# Define a dataset labeling job.
label_task = wf.data.LabelTaskPlaceholder(name="label_task_placeholder_name")

labeling = wf.steps.LabelingStep(
    name="labeling", # Name of the labeling phase. The name contains a maximum of 64 characters,
    including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and must be
    unique in a workflow.
    title="Dataset Labeling", # Title, which defaults to the value of name
    inputs=wf.steps.LabelingInput(name="input_name", data=label_task), # LabelingStep inputs. The
    labeling job object is configured when the workflow is running. You can also use
wf.data.LabelTask(dataset_name="dataset_name", task_name="label_task_name") for the data
    field.
    outputs=wf.steps.LabelingOutput(name="output_name"), # LabelingStep outputs
)

workflow = wf.Workflow(
    name="labeling-step-demo",
    desc="this is a demo workflow",
    steps=[labeling]
)
```

- Scenario 3: Creating a labeling job based on the output of the dataset creation phase

Scenario: The outputs of the dataset creation phase are used as the inputs of the labeling phase.

```
from modelarts import workflow as wf
```

```
# Define parameters of the dataset output path.
dataset_output_path = wf.Placeholder(name="dataset_output_path",
placeholder_type=wf.PlaceholderType.STR, placeholder_format="obs")

# Define the dataset name.
dataset_name = wf.Placeholder(name="dataset_name", placeholder_type=wf.PlaceholderType.STR)

create_dataset = wf.steps.CreateDatasetStep(
```



```
name="create_dataset", # Name of a dataset creation phase. The name contains a maximum of 64
characters, including only letters, digits, underscores (_), and hyphens (-). It must start with a letter
and must be unique in a workflow.
title="Dataset Creation", # Title, which defaults to the value of name
inputs=wf.steps.CreateDatasetInput(name="input_name",
data=wf.data.OBSPlaceholder(name="obs_placeholder_name", object_type="directory")),#
CreateDatasetStep inputs, configured when the workflow is running; the data field can also be
represented by the wf.data.OBSPath(obs_path="fake_obs_path") object.
outputs=wf.steps.CreateDatasetOutput(name="create_dataset_output",
config=wf.data.OBSOutputConfig(obs_path=dataset_output_path)),# CreateDatasetStep outputs
properties=wf.steps.DatasetProperties(
dataset_name=dataset_name, # If the dataset name does not exist, a dataset will be created
using this name. If the dataset name exists, the corresponding dataset will be used.
data_type=wf.data.DataTypeEnum.IMAGE, # Data type of the dataset, for example, image
)
)

# Define the name parameters of the labeling job.
task_name = wf.Placeholder(name="placeholder_name", placeholder_type=wf.PlaceholderType.STR)

labeling = wf.steps.LabelingStep(
name="labeling", # Name of the labeling phase. The name contains a maximum of 64 characters,
including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and must be
unique in a workflow.
title="Dataset Labeling", # Title, which defaults to the value of name
properties=wf.steps.LabelTaskProperties(
task_type=wf.data.LabelTaskTypeEnum.IMAGE_CLASSIFICATION, # Labeling job type, for
example, image classification
task_name=task_name # If the labeling job name does not exist, a job will be created using
this name. If the labeling job name exists, the corresponding job will be used.
),
inputs=wf.steps.LabelingInput(name="input_name",
data=create_dataset.outputs["create_dataset_output"].as_input()), # LabelingStep inputs. The data
source is the outputs of the dataset creation phase.
outputs=wf.steps.LabelingOutput(name="output_name"), # LabelingStep outputs
depend_steps=create_dataset # Preceding dataset creation phase
)
# create_dataset is an instance of wf.steps.CreateDatasetStep. create_dataset_output is the name
field value of wf.steps.CreateDatasetOutput.

workflow = wf.Workflow(
name="labeling-step-demo",
desc="this is a demo workflow",
steps=[create_dataset, labeling]
)
```

4.3.4.3 Creating a Dataset Import Phase

Description

This phase integrates capabilities of the ModelArts dataset module, allowing you to import data to datasets. The dataset import phase is used to import data from a specified path to a dataset or a labeling job. The application scenarios are as follows:

- This phase is used for continuous data update. You can import raw data or labeled data to a labeling job and label the data in the labeling phase.
- Some labeled raw data can be directly imported to a dataset or labeling job, and the dataset with version information can be obtained in the dataset release phase.

Parameter Overview

You can use DatasetImportStep to create a dataset import phase. The following is an example of defining a DatasetImportStep.

Table 4-29 DatasetImportStep

Parameter	Description	Mandatory	Data Type
name	Name of a dataset import phase. The name contains a maximum of 64 characters, including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and must be unique in a workflow.	Yes	str
inputs	Inputs of the dataset import phase.	Yes	DatasetImportInput or DatasetImportInput list
outputs	Outputs of the dataset import phase.	Yes	DatasetImportOutput or DatasetImportOutput list
properties	Configurations for dataset import.	Yes	ImportDataInfo
title	Title for frontend display.	No	str
description	Description of the dataset import phase.	No	str
policy	Phase execution policy.	No	StepPolicy
depend_steps	Dependent phases.	No	Step or step list

Table 4-30 DatasetImportInput

Parameter	Description	Mandatory	Data Type
name	Input name of the dataset import phase. The name can contain a maximum of 64 characters, including only letters, digits, underscores (_), and hyphens (-), and must start with a letter. The input name of a step must be unique.	Yes	str
data	Input data object of the dataset import phase.	Yes	Dataset, OBS, or labeling job object. Currently, only Dataset, DatasetConsumption, DatasetPlaceholder, OBSPath, OBSConsumption, OBSPlaceholder, LabelTask, LabelTaskPlaceholder, LabelTaskConsumption, and DataConsumptionSelector are supported.

Table 4-31 DatasetImportOutput

Parameter	Description	Mandatory	Data Type
name	Output name of the dataset import phase. The name can contain a maximum of 64 characters, including only letters, digits, underscores (_), and hyphens (-), and must start with a letter. The output name of a step must be unique.	Yes	str

Table 4-32 ImportDataInfo

Parameter	Description	Mandatory	Data Type
annotation_form at_config	Configurations of the imported labeling format.	No	AnnotationFormat Config
excluded_labels	Samples with specified labels are not imported.	No	Label list

Parameter	Description	Mandatory	Data Type
import_annotated	Whether to import the labeled samples in the original dataset to the To Be Confirmed tab. The default value is false, indicating that the labeled samples in the original dataset are not imported to the To Be Confirmed tab. The options are as follows: • true: The labeled samples in the original dataset are imported to the To Be Confirmed tab. • false: The labeled samples in the original dataset are not imported to the To Be Confirmed tab.	No	bool
import_annotations	Whether to import labels. The options are as follows: • true: The labels are imported. (Default) • false: The labels are not imported.	No	bool

Parameter	Description	Mandatory	Data Type
import_samples	Whether to import samples. The options are as follows: • true: The samples are imported. (Default) • false: The samples are not imported.	No	bool
import_type	Import mode. The options are as follows: • dir: imported from an OBS path • manifest: imported from a manifest file	No	ImportTypeEnum
included_labels	Samples with specified labels are imported.	No	Label list
label_format	Label format. This parameter is used only for text datasets.	No	LabelFormat

Table 4-33 AnnotationFormatConfig

Parameter	Description	Mandatory	Data Type
format_name	Name of a labeling format	No	AnnotationFormatEnum
parameters	Advanced parameters of the labeling format	No	AnnotationFormatParameters
scene	Labeling scenario, which is optional	No	LabelTaskTypeEnum

Table 4-34 AnnotationFormatParameters

Parameter	Description	Mandatory	Data Type
difficult_only	Whether to import only hard examples. The options are as follows: • true: Only hard examples are imported. • false: All the samples are imported. (Default)	No	bool
included_labels	Samples with specified labels are imported.	No	Label list
label_separator	Separator between labels. By default, the comma (,) is used as the separator. The separator needs to be escaped. The separator can contain only one character, which must be a letter, a digit, or any of the following special characters: !@#\$%^&* _= ?/'/:.,;	No	str
sample_label_separator	Separator between the text and label. By default, the Tab key is used as the separator. The separator needs to be escaped. The separator can contain only one character, which must be a letter, a digit, or any of the following special characters: !@#\$%^&* _= ?/'/:.,;	No	str

Examples

There are three scenarios:

- Scenario 1: Updating a dataset by importing data from a specified path
 - You import labeled data (with label information) in a specified path to a dataset. Then, you can create a dataset release phase to release a version.

Data preparation: Create a dataset on the ModelArts console and upload labeled data to OBS.

```
from modelarts import workflow as wf
# Use DatasetImportStep to import data in a specified path to a dataset and output the dataset.
# Define a dataset.
```

```

dataset = wf.data.DatasetPlaceholder(name="input_dataset")

# Define OBS data.
obs = wf.data.OBSPlaceholder(name = "obs_placeholder_name", object_type = "directory" ) #
object_type must be file or directory.

dataset_import = wf.steps.DatasetImportStep(
    name="data_import", # Name of the dataset import phase. The name contains a maximum
    of 64 characters, including only letters, digits, underscores (_), and hyphens (-). It must start
    with a letter and must be unique in a workflow.
    title="Dataset Import", # Title, which defaults to the value of name
    inputs=[
        wf.steps.DatasetImportInput(name="input_name_1", data=dataset), # The target dataset
        is configured when the workflow is running. You can also use
        wf.data.Dataset(dataset_name="dataset_name") for the data field.
        wf.steps.DatasetImportInput(name="input_name_2", data=obs) # Storage path to the
        imported dataset, configured when the workflow is running. You can also use
        wf.data.OBSPath(obs_path="obs_path") for the data field.
    ],# DatasetImportStep inputs
    outputs=wf.steps.DatasetImportOutput(name="output_name"), # DatasetImportStep outputs
    properties=wf.steps.ImportDataInfo(
        annotation_format_config=[
            wf.steps.AnnotationFormatConfig(
                format_name=wf.steps.AnnotationFormatEnum.MA_IMAGE_CLASSIFICATION_V1, #
                Labeling format of labeled data, for example, image classification
                scene=wf.data.LabelTaskTypeEnum.IMAGE_CLASSIFICATION # Labeling scene
            )
        ]
    )
)

workflow = wf.Workflow(
    name="dataset-import-demo",
    desc="this is a demo workflow",
    steps=[dataset_import]
)

```

- You import unlabeled data in a specified path to a dataset. Then, you can add a labeling phase to label the imported data.

Data preparation: Create a dataset on the ModelArts console and upload unlabeled data to OBS.

```

from modelarts import workflow as wf
# Use DatasetImportStep to import data in a specified path to a dataset and output the dataset.

# Define a dataset.
dataset = wf.data.DatasetPlaceholder(name="input_dataset")

# Define OBS data.
obs = wf.data.OBSPlaceholder(name = "obs_placeholder_name", object_type = "directory" ) #
object_type must be file or directory.

dataset_import = wf.steps.DatasetImportStep(
    name="data_import", # Name of the dataset import phase. The name contains a maximum
    of 64 characters, including only letters, digits, underscores (_), and hyphens (-). It must start
    with a letter and must be unique in a workflow.
    title="Dataset Import", # Title, which defaults to the value of name
    inputs=[
        wf.steps.DatasetImportInput(name="input_name_1", data=dataset), # The target dataset
        is configured when the workflow is running. You can also use
        wf.data.Dataset(dataset_name="dataset_name") for the data field.
        wf.steps.DatasetImportInput(name="input_name_2", data=obs) # Storage path to the
        imported dataset, configured when the workflow is running. You can also use
        wf.data.OBSPath(obs_path="obs_path") for the data field.
    ],# DatasetImportStep inputs
    outputs=wf.steps.DatasetImportOutput(name="output_name"), # DatasetImportStep outputs
)

workflow = wf.Workflow(

```



```
name="dataset-import-demo",
desc="this is a demo workflow",
steps=[dataset_import]
)
```

- Scenario 2: Updating a labeling job by importing data from a specified path
 - You import labeled data in a specified path to a labeling job. Then, you can create a dataset release phase to release a version.

Data preparation: Create a labeling job using a specified dataset and upload the labeled data to OBS.

```
from modelarts import workflow as wf
# Use DatasetImportStep to import data in a specified path to a labeling job and output the
labeling job.

# Define a labeling job.
label_task = wf.data.LabelTaskPlaceholder(name="label_task_placeholder_name")

# Define the OBS data.
obs = wf.data.OBSPlaceholder(name = "obs_placeholder_name", object_type = "directory" ) #
object_type must be file or directory.

dataset_import = wf.steps.DatasetImportStep(
    name="data_import", # Name of the dataset import phase. The name contains a maximum
of 64 characters, including only letters, digits, underscores (_), and hyphens (-). It must start
with a letter and must be unique in a workflow.
    title="Dataset Import", # Title, which defaults to the value of name
    inputs=[
        wf.steps.DatasetImportInput(name="input_name_1", data=label_task), # Labeling job
object, configured when the workflow is running. You can also use
wf.data.LabelTask(dataset_name="dataset_name", task_name="label_task_name") for the
data field.
        wf.steps.DatasetImportInput(name="input_name_2", data=obs) # Storage path to the
imported dataset, configured when the workflow is running. You can also use
wf.data.OBSPath(obs_path="obs_path") for the data field.
    ],# DatasetImportStep inputs
    outputs=wf.steps.DatasetImportOutput(name="output_name"), # DatasetImportStep outputs
    properties=wf.steps.ImportDataInfo(
        annotation_format_config=[
            wf.steps.AnnotationFormatConfig(
                format_name=wf.steps.AnnotationFormatEnum.MA_IMAGE_CLASSIFICATION_V1, #
Labeling format of labeled data, for example, image classification
                scene=wf.data.LabelTaskTypeEnum.IMAGE_CLASSIFICATION # Labeling scene
            )
        ]
    )
)

workflow = wf.Workflow(
    name="dataset-import-demo",
    desc="this is a demo workflow",
    steps=[dataset_import]
)
```

- You import unlabeled data in a specified path to a labeling job. Then, you can add a labeling phase to label the imported data.

Data preparation: Create a labeling job using a specified dataset and upload the unlabeled data to OBS.

```
from modelarts import workflow as wf
# Use DatasetImportStep to import data in a specified path to a labeling job and output the
labeling job.

# Define a labeling job.
label_task = wf.data.LabelTaskPlaceholder(name="label_task_placeholder_name")

# Define the OBS data.
obs = wf.data.OBSPlaceholder(name = "obs_placeholder_name", object_type = "directory" ) #
```

object_type must be **file** or **directory**.

```
dataset_import = wf.steps.DatasetImportStep(  
    name="data_import", # Name of the dataset import phase. The name contains a maximum  
    of 64 characters, including only letters, digits, underscores (_), and hyphens (-). It must start  
    with a letter and must be unique in a workflow.  
    title="Dataset Import", # Title, which defaults to the value of name  
    inputs=[  
        wf.steps.DatasetImportInput(name="input_name_1", data=label_task), # Labeling job  
        object, configured when the workflow is running. You can also use  
        wf.data.LabelTask(dataset_name="dataset_name", task_name="label_task_name") for the  
        data field.  
        wf.steps.DatasetImportInput(name="input_name_2", data=obs) # Storage path to the  
        imported dataset, configured when the workflow is running. You can also use  
        wf.data.OBSPath(obs_path="obs_path") for the data field.  
    ], # DatasetImportStep inputs  
    outputs=wf.steps.DatasetImportOutput(name="output_name"), # DatasetImportStep outputs  
)  
  
workflow = wf.Workflow(  
    name="dataset-import-demo",  
    desc="this is a demo workflow",  
    steps=[dataset_import]  
)
```

- Scenario 3: Creating a dataset import phase using the outputs of the dataset creation phase.

from modelarts import workflow as wf

Use DatasetImportStep to import data in a specified path to a dataset and output the dataset.

Define the OBS data.

obs = wf.data.OBSPlaceholder(name = "obs_placeholder_name", object_type = "directory") #

object_type must be **file** or **directory**.

```
dataset_import = wf.steps.DatasetImportStep(  
    name="data_import", # Name of the dataset import phase. The name contains a maximum of 64  
    characters, including only letters, digits, underscores (_), and hyphens (-). It must start with a letter  
    and must be unique in a workflow.  
    title="Dataset Import", # Title, which defaults to the value of name  
    inputs=[  
        wf.steps.DatasetImportInput(name="input_name_1",  
data=create_dataset.outputs["create_dataset_output"].as_input()), # The outputs of the dataset  
creation phase are used as the inputs of the dataset import phase.  
        wf.steps.DatasetImportInput(name="input_name_2", data=obs) # Storage path to the imported  
dataset, configured when the workflow is running. You can also use  
        wf.data.OBSPath(obs_path="obs_path") for the data field.  
    ], # DatasetImportStep inputs  
    outputs=wf.steps.DatasetImportOutput(name="output_name"), # DatasetImportStep outputs  
    depend_steps=create_dataset # Preceding dataset creation phase  
)  
  
# create_dataset is an instance of wf.steps.CreateDatasetStep. create_dataset_output is the name  
field value of wf.steps.CreateDatasetOutput.  
  
workflow = wf.Workflow(  
    name="dataset-import-demo",  
    desc="this is a demo workflow",  
    steps=[dataset_import]  
)
```

4.3.4.4 Creating a Dataset Release Phase

Description

This phase integrates capabilities of the ModelArts dataset module, enabling automatic dataset version release. The dataset release phase is used to release versions of existing datasets or labeling jobs. Each version is a data snapshot and

can be used for subsequent data source tracing. The application scenarios are as follows:

- After data labeling is completed, a dataset version can be automatically released and used as inputs in subsequent phases.
- When data update is required for model training, you can use the dataset import phase to import data and then use the dataset release phase to release a version for subsequent phases.

Parameter Overview

You can use ReleaseDatasetStep to create a dataset release phase. The following is an example of defining a ReleaseDatasetStep.

Table 4-35 ReleaseDatasetStep

Parameter	Description	Mandatory	Data Type
name	Name of a dataset release phase. The name contains a maximum of 64 characters, including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and must be unique in a workflow.	Yes	str
inputs	Inputs of the dataset release phase.	Yes	ReleaseDatasetInput or ReleaseDatasetInput list
outputs	Outputs of the dataset release phase.	Yes	ReleaseDatasetOutput or ReleaseDatasetOutput list
title	Title for frontend display.	No	str
description	Description of the dataset release phase.	No	str
policy	Phase execution policy.	No	StepPolicy
depend_steps	Dependent phases.	No	Step or step list

Table 4-36 ReleaseDatasetInput

Parameter	Description	Mandatory	Data Type
name	Input name of the dataset release phase. The name can contain a maximum of 64 characters, including only letters, digits, underscores (_), and hyphens (-), and must start with a letter. The input name of a step must be unique.	Yes	str
data	Input data object of the dataset release phase.	Yes	Dataset or labeling job object. Currently, only Dataset, DatasetConsumption, DatasetPlaceholder, LabelTask, LabelTaskPlaceholder, LabelTaskConsumption, and DataConsumptionSelector are supported.

Table 4-37 ReleaseDatasetOutput

Parameter	Description	Mandatory	Data Type
name	Output name of the dataset release phase. The name can contain a maximum of 64 characters, including only letters, digits, underscores (_), and hyphens (-), and must start with a letter. The output name of a step must be unique.	Yes	str
dataset_version_config	Configurations for dataset version release.	Yes	DatasetVersionConfig

Table 4 DatasetVersionConfig

Parameter	Description	Mandatory	Data Type
version_name	Dataset version name. By default, the dataset version is named in ascending order of V001 and V002.	No	str or Placeholder
version_format	Version format, which defaults to Default . You can also set it to CarbonData .	No	str
train_evaluate_sample_ratio	Ratio between the training set and validation set, which defaults to 1.00 . The value ranges from 0 to 1.00. For example, 0.8 indicates the ratio for the training set is 80%, and that for the validation set is 20%.	No	str or Placeholder
clear_hard_property	Whether to clear hard examples. The default value is True .	No	bool or Placeholder
remove_sample_usage	Whether to clear existing usage information of a dataset. The default value is True .	No	bool or Placeholder

Parameter	Description	Mandatory	Data Type
label_task_type	Type of a labeling job. If the input is a dataset, this field is mandatory and is used to specify the labeling scenario of the dataset version. If the input is a labeling job, this field does not need to be configured.	No	LabelTaskTypeEnum The following types are supported: • IMAGE_CLASSIFICATION • OBJECT_DETECTION = 1 • IMAGE_SEGMENTATION • TEXT_CLASSIFICATION • NAMED_ENTITY_RECOGNITION • TEXT_TRIPLE • AUDIO_CLASSIFICATION • SPEECH_CONTENT and SPEECH_SEGMENTATION • TABLE • VIDEO_ANNOTATION
description	Version description.	No	str

 NOTE

If there is no special requirement, use the default values.

Examples

Scenario 1: Releasing a dataset version

Scenario: When data in a dataset is updated, this phase can be used to release a dataset version for subsequent phases to use.

```
from modelarts import workflow as wf
# Use ReleaseDatasetStep to release a version of the input dataset and output the dataset with version information.

# Define a dataset.
dataset = wf.data.DatasetPlaceholder(name="input_dataset")

# Define the split ratio between the training set and validation set
train_ratio = wf.Placeholder(name="placeholder_name", placeholder_type=wf.PlaceholderType.STR, default="0.8")

release_version = wf.steps.ReleaseDatasetStep(
```

```
name="release_dataset", # Name of the dataset release phase. The name contains a maximum of 64
characters, including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and
must be unique in a workflow.
title="Dataset Version Release", # Title, which defaults to the value of name
inputs=wf.steps.ReleaseDatasetInput(name="input_name", data=dataset), # ReleaseDatasetStep inputs.
The dataset object is configured when the workflow is running. You can also use
wf.data.Dataset(dataset_name="dataset_name") for the data field.
outputs=wf.steps.ReleaseDatasetOutput(
    name="output_name",
    dataset_version_config=wf.data.DatasetVersionConfig(
        label_task_type=wf.data.LabelTaskTypeEnum.IMAGE_CLASSIFICATION, # Labeling job type for
dataset version release
        train_evaluate_sample_ratio=train_ratio # Split ratio between the training set and validation set
    )
) # ReleaseDatasetStep outputs
)

workflow = wf.Workflow(
    name="dataset-release-demo",
    desc="this is a demo workflow",
    steps=[release_version]
)
```

Scenario 2: Releasing a labeling job version

When data or labeling information of a labeling job is updated, this phase can be used to release a dataset version for subsequent phases to use.

```
from modelarts import workflow as wf
# Use ReleaseDatasetStep to release a version of the input labeling job and output the dataset with version
information.

# Define a labeling job.
label_task = wf.data.LabelTaskPlaceholder(name="label_task_placeholder_name")

# Define the split ratio between the training set and validation set
train_ratio = wf.Placeholder(name="placeholder_name", placeholder_type=wf.PlaceholderType.STR,
default="0.8")

release_version = wf.steps.ReleaseDatasetStep(
    name="release_dataset", # Name of the dataset release phase. The name contains a maximum of 64
characters, including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and
must be unique in a workflow.
    title="Dataset Version Release", # Title, which defaults to the value of name
    inputs=wf.steps.ReleaseDatasetInput(name="input_name", data=label_task), # ReleaseDatasetStep inputs
The labeling job object is configured when the workflow is running. You can also use
wf.data.LabelTask(dataset_name="dataset_name", task_name="label_task_name") for the data field.
    outputs=wf.steps.ReleaseDatasetOutput(name="output_name",
dataset_version_config=wf.data.DatasetVersionConfig(train_evaluate_sample_ratio=train_ratio)), # Split
ratio between the training set and validation set
)

workflow = wf.Workflow(
    name="dataset-release-demo",
    desc="this is a demo workflow",
    steps=[release_version]
)
```

Scenario 3: Creating a dataset release phase based on the labeling phase

Scenario: The outputs of the labeling phase are used as the inputs of the dataset release phase.

```
from modelarts import workflow as wf
# Use ReleaseDatasetStep to release a version of the input labeling job and output the dataset with version
information.
```

```
# Define the split ratio between the training set and validation set
train_ratio = wf.Placeholder(name="placeholder_name", placeholder_type=wf.PlaceholderType.STR,
default="0.8")

release_version = wf.steps.ReleaseDatasetStep(
    name="release_dataset", # Name of the dataset release phase. The name contains a maximum of 64
characters, including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and
must be unique in a workflow.
    title="Dataset Version Release", # Title, which defaults to the value of name
    inputs=wf.steps.ReleaseDatasetInput(name="input_name",
data=labeling_step.outputs["output_name"].as_input()), # ReleaseDatasetStep inputs
The labeling job object is configured when the workflow is running. You can also use
wf.data.LabelTask(dataset_name="dataset_name", task_name="label_task_name") for the data field.
    outputs=wf.steps.ReleaseDatasetOutput(name="output_name",
dataset_version_config=wf.data.DatasetVersionConfig(train_evaluate_sample_ratio=train_ratio)), # Split
ratio between the training set and validation set
    depend_steps = [labeling_step] # Preceding labeling phase
)
# labeling_step is an instance object of wf.steps.LabelingStep and output_name is the value of the name
field of wf.steps.LabelingOutput.

workflow = wf.Workflow(
    name="dataset-release-demo",
    desc="this is a demo workflow",
    steps=[release_version]
)
```

4.3.4.5 Creating a Training Job Phase

Description

This phase defines the algorithm, input, and output of a job for data processing, model training, or model evaluation. The application scenarios are as follows:

- Data preprocessing such as image enhancement and noise reduction
- Model training for object detection and image classification

Parameter Overview

You can use JobStep to create a job phase. The following is an example of defining a JobStep.

Table 4-38 JobStep

Parameter	Description	Mandatory	Data Type
name	Name of a job phase. The name contains a maximum of 64 characters, including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and must be unique in a workflow.	Yes	str
algorithm	Algorithm object.	Yes	• BaseAlgorithm • Algorithm • AIGalleryAlgorithm
spec	Job specifications.	Yes	JobSpec
inputs	Inputs of a job phase.	Yes	JobInput or JobInput list
outputs	Outputs of a job phase.	Yes	JobOutput or JobOutput list
title	Title for frontend display.	No	str
description	Description of a job phase.	No	str
policy	Phase execution policy.	No	StepPolicy
depend_steps	Dependent phases.	No	Step or step list

Table 4-39 JobInput

Parameter	Description	Mandatory	Data Type
name	Input name of the job phase. The name can contain a maximum of 64 characters, including only letters, digits, underscores (_), and hyphens (-), and must start with a letter. The input name of a step must be unique.	Yes	str
data	Input data object of a job phase.	Yes	Dataset or OBS object. Currently, only Dataset, DatasetPlaceholder, DatasetConsumption, OBSPath, OBSConsumption, OBSPlaceholder, and DataConsumption Selector are supported.

Table 4-40 JobOutput

Parameter	Description	Mandatory	Data Type
name	Output name of the job phase. The name can contain a maximum of 64 characters, including only letters, digits, underscores (_), and hyphens (-), and must start with a letter. The output name of a step must be unique.	Yes	str

Parameter	Description	Mandatory	Data Type
obs_config	OBS output configuration.	No	OBSOutputConfig
model_config	Model output configuration.	No	ModelConfig
metrics_config	Metrics configuration.	No	MetricsConfig

Table 4-41 OBSOutputConfig

Parameter	Description	Mandatory	Data Type
obs_path	Existing OBS directory	Yes	str, Placeholder, Storage
metric_file	Name of the file that stores metric information	No	str, Placeholder

Table 4-42 BaseAlgorithm

Parameter	Description	Mandatory	Data Type
id	Algorithm ID	No	str
subscription_id	Subscription ID of the subscribed algorithm	No	str
item_version_id	Version ID of the subscribed algorithm	No	str
code_dir	Code directory	No	str, Placeholder, Storage
boot_file	Boot file	No	str, Placeholder, Storage
command	Boot command	No	str, Placeholder
parameters	Algorithm hyperparameters	No	AlgorithmParameters list
engine	Information about the image used by the job	No	JobEngine
environments	Environment variables	No	dict

Table 4-43 Algorithm

Parameter	Description	Mandatory	Data Type
algorithm_id	Algorithm ID	Yes	str
parameters	Algorithm hyperparameters	No	List of algorithm parameters

Table 4-44 AIGalleryAlgorithm

Parameter	Description	Mandatory	Data Type
subscription_id	Subscription ID of the subscribed algorithm	Yes	str
item_version_id	Version ID of the subscribed algorithm	Yes	str
parameters	Algorithm hyperparameters	No	List of algorithm parameters

Table 4-45 AlgorithmParameters

Parameter	Description	Mandatory	Data Type
name	Name of an algorithm hyperparameter	Yes	str
value	Value of an algorithm hyperparameter	Yes	int, bool, float, str, Placeholder, Storage

Table 4-46 JobEngine

Parameter	Description	Mandatory	Data Type
engine_id	Image ID	No	str, Placeholder
engine_name	Image name	No	str, Placeholder
engine_version	Image version	No	str, Placeholder
image_url	Image URL	No	str, Placeholder

Table 4-47 JobSpec

Parameter	Description	Mandatory	Data Type
resource	Resource information	Yes	JobResource
log_export_path	Log output path	No	LogExportPath
schedule_policy	Job scheduling policy	No	SchedulePolicy
volumes	Information about the file system mounted to the job	No	list[Volume]

Table 4-48 JobResource

Parameter	Description	Mandatory	Data Type
flavor	Resource flavor.	Yes	Placeholder
node_count	Number of nodes. The default value is 1 . If there are multiple nodes, distributed training is supported.	No	int, Placeholder

Table 4-49 SchedulePolicy

Parameter	Description	Mandatory	Data Type
priority	Job scheduling priority. The value can only be 1 , 2 , or 3 , indicating low, medium, and high priorities, respectively.	Yes	int, Placeholder

Table 4-50 Volume

Parameter	Description	Mandatory	Data Type
nfs	NFS file system object. In a volume object, only one of nfs , pacific , and pfs can be configured.	No	NFS
pacific	Pacific file system object. In a volume object, only one of nfs , pacific , and pfs can be configured.	No	Placeholder
pfs	OBS parallel file system object. In a volume object, only one of nfs , pacific , and pfs can be configured.	No	PFS, Placeholder

Table 4-51 NFS

Parameter	Description	Mandatory	Data Type
nfs_server_path	Service address of the NFS file system.	Yes	str, Placeholder
local_path	Path mounted to the container.	Yes	str, Placeholder
read_only	Indicates if the mount mode is set to read-only.	No	bool, Placeholder

Table 4-52 PFS

Parameter	Description	Mandatory	Data Type
pfs_path	Path of the parallel file system	Yes	str, Placeholder
local_path	Path mounted to the container	Yes	str, Placeholder

Obtaining Resource Flavors

Before creating a job phase, perform the following operations to obtain supported training flavors and engines:

- Import packages.

```
from modelarts.session import Session
from modelarts.estimatorV2 import TrainingJob
from modelarts.workflow.client.job_client import JobClient
```

- Initialize a session.

```
# If you develop a workflow in a local IDEA, initialize a session as follows:
# Hardcoded or plaintext AK/SK is risky. For security, encrypt your AK/SK and store them in the
# configuration file or environment variables.
# In this example, the AK/SK are stored in environment variables for identity authentication. Before
# running this example, set environment variables HUAWEICLOUD_SDK_AK and
# HUAWEICLOUD_SDK_SK.
__AK = os.environ["HUAWEICLOUD_SDK_AK"]
__SK = os.environ["HUAWEICLOUD_SDK_SK"]
# Decrypt the information if it is encrypted.
session = Session(
    access_key=__AK, # AK information of your account
    secret_key=__SK, # SK information of your account
    region_name="****", # Region to which your account belongs
    project_id="****" # Project ID of your account
)

# If you develop a workflow in a notebook environment, initialize a session:
session = Session()
```

- Obtain public resource pools.

```
# Obtain the specification list of public resource pools.
spec_list = TrainingJob(session).get_train_instance_types(session) # A list is returned. You can
# download it.
print(spec_list)
```

- Obtain dedicated resource pools.

```
# Obtain the list of running dedicated resource pools.
pool_list = JobClient(session).get_pool_list() # A list of dedicated resource pools is returned.
pool_id_list = JobClient(session).get_pool_id_list() # An ID list of dedicated resource pools is returned.
# The following lists the flavor IDs of dedicated resource pools. Select one as required.
modelarts.pool.visual.xlarge (1 card)
modelarts.pool.visual.2xlarge (2 cards)
modelarts.pool.visual.4xlarge (4 cards)
modelarts.pool.visual.8xlarge (8 cards)
```

- Obtain engine types.

```
# Obtain engine types.
engine_dict = TrainingJob(session).get_engine_list(session) # A dictionary is returned. You can
# download it.
print(engine_dict)
```

Examples

There are seven scenarios:

- Using an algorithm subscribed to in AI Gallery
- Using an algorithm in Algorithm Management
- Using a custom algorithm (code directory+boot file+official image)
- Using a custom algorithm (code directory+boot command+official image)
- Creating a job phase based on the dataset release phase
- Job phase with visualization
- Using the DataSelector object as the input, which supports OBS or datasets

Using an Algorithm Subscribed from AI Gallery

```
from modelarts import workflow as wf

# Create an OutputStorage object to centrally manage training output directories.
storage = wf.data.OutputStorage(name="storage_name", title="title_info", description="description_info") #
Only name is mandatory.

# Define an input dataset.
dataset = wf.data.DatasetPlaceholder(name="input_dataset")

# Use JobStep to define a training phase. Use a dataset as the input, and use OBS to store the output.
job_step = wf.steps.JobStep(
    name="training_job", # Name of a training phase. The name contains a maximum of 64 characters,
    including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and must be
    unique in a workflow.
    title="Image Classification Training", # Title, which defaults to the value of name
    algorithm=wf.AIGalleryAlgorithm(
        subscription_id="subscription_id", # Algorithm subscription ID. You can also enter the version number.
        item_version_id="item_version_id", # Algorithm version ID. You can also enter the version number
        instead.
        parameters=[
            wf.AlgorithmParameters(
                name="parameter_name",
                value=wf.Placeholder(name="parameter_name", placeholder_type=wf.PlaceholderType.STR,
                default="fake_value",description="description_info")
            ) # Algorithm hyperparameters are represented using placeholders, which can be integer, bool,
            float, or string.
        ],
    ), # Algorithm used for training. An algorithm subscribed to in AI Gallery is used in this example. If the
    value of an algorithm hyperparameter does not need to be changed, you do not need to configure the
    hyperparameter in parameters. Hyperparameter values will be automatically filled.

    inputs=wf.steps.JobInput(name="data_url", data=dataset), # JobStep inputs are configured when the
    workflow is running. You can also use wf.data.Dataset(dataset_name="fake_dataset_name",
    version_name="fake_version_name") for the data field.
    outputs=wf.steps.JobOutput(name="train_url",
    obs_config=wf.data.OBSOutputConfig(obs_path=storage.join("directory_path"))), # JobStep outputs
    spec=wf.steps.JobSpec(
        resource=wf.steps.JobResource(
            flavor=wf.Placeholder(name="train_flavor", placeholder_type=wf.PlaceholderType.JSON,
            description="Training flavor")

        )
    )# Training flavors
)

workflow = wf.Workflow(
    name="job-step-demo",
    desc="this is a demo workflow",
    steps=[job_step],
    storages=[storage]
)
```

Using an Algorithm in Algorithm Management

```
from modelarts import workflow as wf

# Create an OutputStorage object to centrally manage training output directories.
storage = wf.data.OutputStorage(name="storage_name", title="title_info", description="description_info") #
Only name is mandatory.

# Define an input dataset.
dataset = wf.data.DatasetPlaceholder(name="input_dataset")

# Use JobStep to define a training phase. Use a dataset as the input, and use OBS to store the output.
job_step = wf.steps.JobStep(
    name="training_job", # Name of a training phase. The name contains a maximum of 64 characters,
    including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and must be
    unique in a workflow.
```



```
title="Image Classification Training", # Title, which defaults to the value of name
algorithm=wf.Algorithm(
    algorithm_id="algorithm_id", # Algorithm ID
    parameters=[
        wf.AlgorithmParameters(
            name="parameter_name",
            value=wf.Placeholder(name="parameter_name", placeholder_type=wf.PlaceholderType.STR,
default="fake_value",description="description_info")
        ) # Algorithm hyperparameters are represented using placeholders, which can be integer, bool,
float, or string.
    ]
), # Algorithm used for training. An algorithm from Algorithm Management is used in this example. If
the value of an algorithm hyperparameter does not need to be changed, you do not need to configure the
hyperparameter in parameters. Hyperparameter values will be automatically filled.

inputs=wf.steps.JobInput(name="data_url", data=dataset), # JobStep inputs are configured when the
workflow is running. You can also use wf.data.Dataset(dataset_name="fake_dataset_name",
version_name="fake_version_name") for the data field.
outputs=wf.steps.JobOutput(name="train_url",
obs_config=wf.data.OBSOutputConfig(obs_path=storage.join("directory_path"))), # JobStep outputs
spec=wf.steps.JobSpec(
    resource=wf.steps.JobResource(
        flavor=wf.Placeholder(name="train_flavor", placeholder_type=wf.PlaceholderType.JSON,
description="Training flavor")
    )
) # Training flavors
)

workflow = wf.Workflow(
    name="job-step-demo",
    desc="this is a demo workflow",
    steps=[job_step],
    storages=[storage]
)
```

Using a Custom Algorithm (Code Directory + Boot File + Official Image)

```
from modelarts import workflow as wf

# Create an OutputStorage object to centrally manage training output directories.
storage = wf.data.OutputStorage(name="storage_name", title="title_info", description="description_info") #
Only name is mandatory.

# Define an input dataset.
dataset = wf.data.DatasetPlaceholder(name="input_dataset")

# Use JobStep to define a training phase. Use a dataset as the input, and use OBS to store the output.
job_step = wf.steps.JobStep(
    name="training_job", # Name of a training phase. The name contains a maximum of 64 characters,
including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and must be
unique in a workflow.
    title="Image Classification Training", # Title, which defaults to the value of name
    algorithm=wf.BaseAlgorithm(
        code_dir="fake_code_dir", # Code directory
        boot_file="fake_boot_file", # Boot file path, which must be in the code directory
        engine=wf.steps.JobEngine(engine_name="fake_engine_name",
engine_version="fake_engine_version"), # Name and version of the official image

        parameters=[
            wf.AlgorithmParameters(
                name="parameter_name",
                value=wf.Placeholder(name="parameter_name", placeholder_type=wf.PlaceholderType.STR,
default="fake_value",description="description_info")
            ) # Algorithm hyperparameters are represented using placeholders, which can be integer, bool,
float, or string.
        ]
    ), # The custom algorithm is implemented using the code directory, boot file, and official image.
```

```
        inputs=wf.steps.JobInput(name="data_url", data=dataset), # JobStep inputs are configured when the
        workflow is running. You can also use wf.data.Dataset(dataset_name="fake_dataset_name",
        version_name="fake_version_name") for the data field.
        outputs=wf.steps.JobOutput(name="train_url",
        obs_config=wf.data.OBSOutputConfig(obs_path=storage.join("directory_path"))), # JobStep outputs
        spec=wf.steps.JobSpec(
            resource=wf.steps.JobResource(
                flavor=wf.Placeholder(name="train_flavor", placeholder_type=wf.PlaceholderType.JSON,
                description="Training flavor")
            )
        ) # Training flavors
    )

workflow = wf.Workflow(
    name="job-step-demo",
    desc="this is a demo workflow",
    steps=[job_step],
    storages=[storage]
)
```

Using a Custom Algorithm (Code Directory + Boot Command + Custom Image)

```
from modelarts import workflow as wf
```

```
# Create an OutputStorage object to centrally manage training output directories.
storage = wf.data.OutputStorage(name="storage_name", title="title_info", description="description_info") #
Only name is mandatory.
```

```
# Define an input dataset.
dataset = wf.data.DatasetPlaceholder(name="input_dataset")
```

```
# Use JobStep to define a training phase. Use a dataset as the input, and use OBS to store the output.
job_step = wf.steps.JobStep(
    name="training_job", # Name of a training phase. The name contains a maximum of 64 characters,
    including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and must be
    unique in a workflow.
    title="Image Classification Training", # Title, which defaults to the value of name
    algorithm=wf.BaseAlgorithm(
        code_dir="fake_code_dir", # Code directory
        command="fake_command", # Boot command
        engine=wf.steps.JobEngine(image_url="fake_image_url"), # Custom image URL, in the format of
        Organization name/Image name:Version name. Do not contain the domain name; If image_url is required
        to be configurable in the running state, use the following: image_url=wf.Placeholder(name="image_url",
        placeholder_type=wf.PlaceholderType.STR, placeholder_format="swr", description="Custom image")
        parameters=[
            wf.AlgorithmParameters(
                name="parameter_name",
                value=wf.Placeholder(name="parameter_name", placeholder_type=wf.PlaceholderType.STR,
                default="fake_value",description="description_info")
            ) # Algorithm hyperparameters are represented using placeholders, which can be integer, bool,
            float, or string.
        ]
    ), The custom algorithm is implemented using the code directory, boot command, and custom image.
```

```
        inputs=wf.steps.JobInput(name="data_url", data=dataset), # JobStep inputs are configured when the
        workflow is running. You can also use wf.data.Dataset(dataset_name="fake_dataset_name",
        version_name="fake_version_name") for the data field.
        outputs=wf.steps.JobOutput(name="train_url",
        obs_config=wf.data.OBSOutputConfig(obs_path=storage.join("directory_path"))), # JobStep outputs
        spec=wf.steps.JobSpec(
            resource=wf.steps.JobResource(
                flavor=wf.Placeholder(name="train_flavor", placeholder_type=wf.PlaceholderType.JSON,
                description="Training flavor")
            )
        ) # Training flavors
    )
```

```
workflow = wf.Workflow(  
    name="job-step-demo",  
    desc="this is a demo workflow",  
    steps=[job_step],  
    storages=[storage]  
)
```

NOTE

The preceding four methods use a dataset as the input. If you want to use an OBS path as the input, set **data** of **JobInput** to **data=wf.data.OBSPlaceholder(name="obs_placeholder_name", object_type="directory")** or **data=wf.data.OBSPath(obs_path="fake_obs_path")**.

In addition, you can specify a dataset or OBS path when creating a workflow to reduce configuration operations and facilitate debugging in the development state. You are advised to use placeholders to create a workflow you want to publish to the running state or AI Gallery. In this case, you can configure parameters before workflow execution.

Creating a Job Phase Based on the Dataset Release Phase

Scenario: The output of the dataset release phase is used as the input of the job phase.

```
from modelarts import workflow as wf  
  
# Define the dataset object.  
dataset = wf.data.DatasetPlaceholder(name="input_dataset")  
  
# Define the split ratio between the training set and validation set  
train_ratio = wf.Placeholder(name="placeholder_name", placeholder_type=wf.PlaceholderType.STR,  
default="0.8")  
  
release_version_step = wf.steps.ReleaseDatasetStep(  
    name="release_dataset", # Name of the dataset release phase. The name contains a maximum of 64  
                           # characters, including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and  
                           # must be unique in a workflow.  
    title="Dataset Version Release", # Title, which defaults to the value of name  
    inputs=wf.steps.ReleaseDatasetInput(name="input_name", data=dataset), # ReleaseDatasetStep inputs.  
    # The dataset object is configured when the workflow is running. You can also use  
    # wf.data.Dataset(dataset_name="dataset_name") for the data field.  
    outputs=wf.steps.ReleaseDatasetOutput(  
        name="output_name",  
        dataset_version_config=wf.data.DatasetVersionConfig(  
            label_task_type=wf.data.LabelTaskTypeEnum.IMAGE_CLASSIFICATION, # Labeling job type for  
dataset version release  
            train_evaluate_sample_ratio=train_ratio # Split ratio between the training set and validation set  
        )  
    ) # ReleaseDatasetStep outputs  
)  
  
# Create an OutputStorage object to centrally manage training output directories.  
storage = wf.data.OutputStorage(name="storage_name", title="title_info", description="description_info") #  
Only name is mandatory.  
  
# Use JobStep to define a training phase. Use a dataset as the input, and use OBS to store the output.  
job_step = wf.steps.JobStep(  
    name="training_job", # Name of a training phase. The name contains a maximum of 64 characters,  
                        # including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and must be  
                        # unique in a workflow.  
    title="Image Classification Training", # Title, which defaults to the value of name  
    algorithm=wf.AIGalleryAlgorithm(  
        subscription_id="subscription_id", # Subscription ID of the subscribed algorithm  
        item_version_id="item_version_id", # Version ID of the subscribed algorithm  
        parameters=[  
            wf.AlgorithmParameters(  
                name="parameter_name",  
                value=wf.Placeholder(name="parameter_name", placeholder_type=wf.PlaceholderType.STR,  
default="fake_value",description="description_info")
```

```
    ) # Algorithm hyperparameters are represented using placeholders, which can be integer, bool,
float, or string.
    ]
    ), # Algorithm used for training. An algorithm subscribed to in AI Gallery is used in this example. If the
value of an algorithm hyperparameter does not need to be changed, you do not need to configure the
hyperparameter in parameters. Hyperparameter values will be automatically filled.

    inputs=wf.steps.JobInput(name="data_url",
data=release_version_step.outputs["output_name"].as_input()), # The output of the dataset release phase is
used as the input of JobStep.
    outputs=wf.steps.JobOutput(name="train_url",
obs_config=wf.data.OBSOutputConfig(obs_path=storage.join("directory_path"))), # JobStep outputs
spec=wf.steps.JobSpec(
    resource=wf.steps.JobResource(
        flavor=wf.Placeholder(name="train_flavor", placeholder_type=wf.PlaceholderType.JSON,
description="Training flavor")
    )
), # Training flavors
depend_steps=release_version_step # Preceding dataset release phase
)
# release_version_step is an instance object of wf.steps.ReleaseDatasetStep and output_name is the
value of the name field of wf.steps.ReleaseDatasetOutput.

workflow = wf.Workflow(
    name="job-step-demo",
    desc="this is a demo workflow",
    steps=[release_version_step, job_step],
    storages=[storage]
)
```

Job Phase With Visualization

Phase visualization enables you to view the metrics generated by your workflows in real time. You can also display the external disks of each phase separately. To use phase visualization, you need to add and configure an output for showing metrics through the MetricsConfig object, based on the original job phase.

Table 4-53 MetricsConfig

Parameter	Description	Mandatory	Data Type
metric_files	Metric files. Supported element types: str, Placeholder, and Storage.	Yes	list
realtime_visualization	Whether to display the output metrics in real time. The default value is False .	No	bool
visualization	Whether to display visualization phases separately. The default value is True .	No	bool

The output metrics file must contain standard JSON data with a maximum size of 1 MB. The data formats must match the supported ones.

- Key-value pair data

```
[
  {
    "key": "loss",
    "title": "loss",
    "type": "float",
    "data": {
      "value": 1.2
    }
  },
  {
    "key": "accuracy",
    "title": "accuracy",
    "type": "float",
    "data": {
      "value": 1.6
    }
  }
]
```

- Line chart data

```
[
  {
    "key": "metric",
    "title": "metric",
    "type": "line chart",
    "data": {
      "x_axis": [
        {
          "title": "step/epoch",
          "value": [
            1,
            2,
            3
          ]
        }
      ],
      "y_axis": [
        {
          "title": "value",
          "value": [
            0.5,
            0.4,
            0.3
          ]
        }
      ]
    }
  }
]
```

- Histogram data

```
[
  {
    "key": "metric",
    "title": "metric",
    "type": "histogram",
    "data": {
      "x_axis": [
        {
          "title": "step/epoch",
          "value": [
            1,
            2,
            3
          ]
        }
      ]
    }
  }
]
```

```

    ],
    "y_axis": [
      {
        "title": "value",
        "value": [
          0.5,
          0.4,
          0.3
        ]
      }
    ]
  }
}
]

```

- Confusion matrix

```

[
  {
    "key": "confusion_matrix",
    "title": "confusion_matrix",
    "type": "table",
    "data": {
      "cell_value": [
        [
          1,
          2
        ],
        [
          2,
          3
        ]
      ],
      "col_labels": {
        "title": "labels",
        "value": [
          "daisy",
          "dandelion"
        ]
      },
      "row_labels": {
        "title": "predictions",
        "value": [
          "daisy",
          "dandelion"
        ]
      }
    }
  }
]

```

- One-dimensional table

```

[
  {
    "key": "Application Evaluation Results",
    "title": "Application Evaluation Results",
    "type": "one-dimensional-table",
    "data": {
      "cell_value": [
        [
          10,
          2,
          0.5
        ]
      ],
      "labels": [
        "samples",
        "maxResTine",
        "p99"
      ]
    }
  }
]

```

```

    }
]

Example:
from modelarts import workflow as wf

# Create a Storage object to centrally manage training output directories.
storage = wf.data.Storage(name="storage_name", title="title_info", description="description_info",
with_execution_id=True, create_dir=True) # Only name is mandatory.

# Define an input dataset.
dataset = wf.data.DatasetPlaceholder(name="input_dataset")

# Use JobStep to define a training phase. Use a dataset as the input, and use OBS to store the output.
job_step = wf.steps.JobStep(
    name="training_job", # Name of a training phase. The name contains a maximum of 64
    characters, including only letters, digits, underscores (_), and hyphens (-). It must start with a letter
    and must be unique in a workflow.
    title="Image Classification Training", # Title, which defaults to the value of name
    algorithm=wf.AIGalleryAlgorithm(
        subscription_id="subscription_id", # Subscription ID of the subscribed algorithm
        item_version_id="item_version_id", # Algorithm version ID. You can also enter the version
        number instead.
        parameters=[
            wf.AlgorithmParameters(
                name="parameter_name",
                value=wf.Placeholder(name="parameter_name", placeholder_type=wf.PlaceholderType.STR,
                default="fake_value", description="description_info")
            ) # Algorithm hyperparameters are represented using placeholders, which can be integer,
            bool, float, or string.
        ]

    ), # Algorithm used for training. An algorithm subscribed to in AI Gallery is used in this example. If
    the value of an algorithm hyperparameter does not need to be changed, you do not need to
    configure the hyperparameter in parameters. Hyperparameter values will be automatically filled.

    inputs=wf.steps.JobInput(name="data_url", data=dataset), # JobStep inputs are configured when
    the workflow is running. You can also use wf.data.Dataset(dataset_name="fake_dataset_name",
    version_name="fake_version_name") for the data field.
    outputs=[
        wf.steps.JobOutput(name="train_url",
        obs_config=wf.data.OBSOutputConfig(obs_path=storage.join("directory_path"))), # JobStep outputs
        wf.steps.JobOutput(name="metrics_output",
        metrics_config=wf.data.MetricsConfig(metric_files=storage.join("directory_path/metrics.json",
        create_dir=False))) # Metrics are output to the configured path by the job script.
    ],
    spec=wf.steps.JobSpec(
        resource=wf.steps.JobResource(
            flavor=wf.Placeholder(name="train_flavor", placeholder_type=wf.PlaceholderType.JSON,
            description="Training flavor")

        )
    ) # Training flavors
)

workflow = wf.Workflow(
    name="job-step-demo",
    desc="this is a demo workflow",
    steps=[job_step],
    storages=[storage]
)

```

 NOTE

Workflow does not automatically retrieve the metrics produced by training. You need to extract the metrics from the algorithm code, create the **metrics.json** file in the required data format, and upload the file to the OBS path specified in MetricsConfig. Workflow only reads, renders, and displays the data.

Using the DataSelector Object as the Input, Which Supports OBS or Datasets

You can use this method when you can choose the input type. The DataSelector object allows you to select either a dataset object or an OBS object as the training input. Here is a code sample:

```
from modelarts import workflow as wf

# Create an OutputStorage object to centrally manage training output directories.
storage = wf.data.OutputStorage(name="storage_name", title="title_info", description="description_info") #
Only name is mandatory.

# Define the DataSelector object.
data_selector = wf.data.DataSelector(name="input_data", data_type_list=["dataset", "obs"])

# Use JobStep to define a training phase. Use a dataset as the input, and use OBS to store the output.
job_step = wf.steps.JobStep(
    name="training_job", # Name of a training phase. The name contains a maximum of 64 characters,
    including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and must be
    unique in a workflow.
    title="Image Classification Training", # Title, which defaults to the value of name
    algorithm=wf.AIGalleryAlgorithm(
        subscription_id="subscription_id", # Algorithm subscription ID. You can also enter the version number.
        item_version_id="item_version_id", # Algorithm version ID. You can also enter the version number
        instead.
        parameters=[
            wf.AlgorithmParameters(
                name="parameter_name",
                value=wf.Placeholder(name="parameter_name", placeholder_type=wf.PlaceholderType.STR,
                default="fake_value",description="description_info")
            ) # Algorithm hyperparameters are represented using placeholders, which can be integer, bool,
            float, or string.
        ],
        ), # Algorithm used for training. An algorithm subscribed to in AI Gallery is used in this example. If the
        value of an algorithm hyperparameter does not need to be changed, you do not need to configure the
        hyperparameter in parameters. Hyperparameter values will be automatically filled.

    inputs=wf.steps.JobInput(name="data_url", data=data_selector), # JobStep inputs are configured when
    the workflow is running. You can choose OBS or datasets as the input.
    outputs=wf.steps.JobOutput(name="train_url",
    obs_config=wf.data.OBSOutputConfig(obs_path=storage.join("directory_path"))), # JobStep outputs
    spec=wf.steps.JobSpec(
        resource=wf.steps.JobResource(
            flavor=wf.Placeholder(name="train_flavor", placeholder_type=wf.PlaceholderType.JSON,
            description="Training flavor")

        )
    ) # Training flavors
)

workflow = wf.Workflow(
    name="job-step-demo",
    desc="this is a demo workflow",
    steps=[job_step],
    storages=[storage]
)
```

NOTE

When using DataSelector as the input, ensure that the algorithm input supports both datasets and OBS.

4.3.4.6 Creating a Model Registration Phase

Description

This phase integrates capabilities of ModelArts AI application management. This enables trained models to be registered in AI Application Management for service deployment and update. The application scenarios are as follows:

- Registering models trained from ModelArts training jobs
- Registering models from custom images

Parameter Overview

You can use ModelStep to create a model registration phase. The following is an example of defining a ModelStep.

Table 4-54 ModelStep

Parameter	Description	Mandatory	Data Type
name	Name of a model registration phase. The name contains a maximum of 64 characters, including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and must be unique in a workflow.	Yes	str
inputs	Inputs of the model registration phase.	No	ModelInput or ModelInput list
outputs	Outputs of the model registration phase.	Yes	ModelOutput or ModelOutput list
title	Title for frontend display.	No	str
description	Description of the model registration phase.	No	str
policy	Phase execution policy.	No	StepPolicy
depend_steps	Dependent phases.	No	Step or step list

Table 4-55 ModelInput

Parameter	Description	Mandatory	Data Type
name	Input name of the model registration phase. The name can contain a maximum of 64 characters, including only letters, digits, underscores (_), and hyphens (-), and must start with a letter. The input name of a step must be unique.	Yes	str
data	Input data object of the model registration phase.	Yes	OBS, SWR, or subscribed model object. Currently, only OBSPath, SWRImage, OBSConsumption, OBSPlaceholder, SWRImagePlaceholder, DataConsumption Selector, and GalleryModel are supported.

Table 4-56 ModelOutput

Parameter	Description	Mandatory	Data Type
name	Output name of the model registration phase. The name can contain a maximum of 64 characters, including only letters, digits, underscores (_), and hyphens (-), and must start with a letter. The output name of a step must be unique.	Yes	str
model_config	Configurations for model registration.	Yes	ModelConfig

Table 4-57 ModelConfig

Parameter	Description	Mandatory	Data Type
model_type	Model type. Supported types: TensorFlow , MXNet , Caffe , Spark_Mllib , Scikit_Learn , XGBoost , Image , PyTorch , Template , and Custom . The default value is TensorFlow .	Yes	str
model_name	Model name. Enter 1 to 64 characters. Only letters, digits, hyphens (-), and underscores (_) are allowed.	No	str, Placeholder
model_version	Model version in the format of <i>Digit.Digit.Digit</i> . The value range of the digits is [1, 99]. If this parameter is left blank, the version number automatically increases. CAUTION No part of the version number can start with 0. For example, 01.01.01 is not allowed.	No	str, Placeholder
runtime	Model runtime environment. The options of runtime are the same as those of model_type .	No	str, Placeholder
description	Model description that consists of 1 to 100 characters. The following special characters cannot be contained: &!'"<>=	No	str
execution_code	OBS path for storing the execution code. By default, this parameter is left blank. The name of the execution code file is fixed to customize_service.py . The inference code file must be stored in the model directory. This parameter is left blank. The system can automatically identify the inference code in the model directory.	No	str
dependencies	Package required for the inference code and model. By default, this parameter is left blank. It is read from the configuration file.	No	str
model_metrics	Model precision, which is read from the configuration file.	No	str

Parameter	Description	Mandatory	Data Type
apis	All apis input and output parameters of a model (optional), which are parsed from the configuration file.	No	str
initial_config	Model configuration information.	No	dict
template	Template configuration items. This parameter is mandatory when model_type is set to Template .	No	Template
dynamic_load_mode	Dynamic loading mode. Currently, only Single is supported.	No	str, Placeholder
prebuild	Whether the model is prebuilt. The default value is False .	No	bool, Placeholder
install_type	Model installation type. The value can be real_time , edge , batch . If this parameter is left blank, all types are supported by default.	No	list[str]

Table 4-58 Template

Parameter	Description	Mandatory	Data Type
template_id	ID of the used template. The template has a built-in input and output mode.	Yes	str, Placeholder
infer_format	Input and output mode. When this parameter is used, the input and output mode built in the template does not take effect.	No	str, Placeholder
template_inputs	Template input configuration, specifying the source path for configuring a model	Yes	list of TemplateInputs object

Table 4-59 TemplateInputs

Parameter	Description	Mandatory	Data Type
input_id	Input item ID, which is obtained from the template details.	Yes	str, Placeholder

Parameter	Description	Mandatory	Data Type
input	Template input path, which can be an OBS file path or OBS directory path. When you use a template with multiple input items to create a model, if the target paths input_properties specified in the template are the same, the OBS directory or OBS file name entered here must be unique to prevent files from being overwritten.	Yes	str, Placeholder, Storage

Examples

There are six scenarios:

- Registering models output by JobStep
- Registering a model using OBS data
- Registering a model using a template
- Registering a model using a custom image
- Registering a model using a custom image and OBS
- Registering a model using a subscribed model and OBS

Registering a Model from a Training Job (Model Source: JobStep Output)

```
import modelarts.workflow as wf

# Create an OutputStorage object to centrally manage training output directories.
storage = wf.data.OutputStorage(name="storage_name", title="title_info", description="description_info") #
Only name is mandatory.

# Define an input dataset.
dataset = wf.data.DatasetPlaceholder(name="input_dataset")

# Use JobStep to define a training phase. Use a dataset as the input, and use OBS to store the output.
job_step = wf.steps.JobStep(
    name="training_job", # Name of a training phase. The name contains a maximum of 64 characters,
    including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and must be
    unique in a workflow.
    title="Image Classification Training", # Title, which defaults to the value of name
    algorithm=wf.AIGalleryAlgorithm(
        subscription_id="subscription_id", # Algorithm subscription ID. You can also enter the version number.
        item_version_id="item_version_id", # Algorithm version ID. You can also enter the version number
        instead.
        parameters=[
            wf.AlgorithmParameters(
                name="parameter_name",
                value=wf.Placeholder(name="parameter_name", placeholder_type=wf.PlaceholderType.STR,
                default="fake_value",description="description_info")
            ) # Algorithm hyperparameters are represented using placeholders, which can be integer, bool,
            float, or string.
        ],
    ), # Algorithm used for training. An algorithm subscribed to in AI Gallery is used in this example. If the
    value of an algorithm hyperparameter does not need to be changed, you do not need to configure the
    hyperparameter in parameters. Hyperparameter values will be automatically filled.
```

```
inputs=wf.steps.JobInput(name="data_url", data=dataset), # JobStep inputs are configured when the
workflow is running. You can also use wf.data.Dataset(dataset_name="fake_dataset_name",
version_name="fake_version_name") for the data field.
outputs=wf.steps.JobOutput(name="train_url",
obs_config=wf.data.OBSOutputConfig(obs_path=storage.join("directory_path"))), # JobStep outputs
spec=wf.steps.JobSpec(
    resource=wf.steps.JobResource(
        flavor=wf.Placeholder(name="train_flavor", placeholder_type=wf.PlaceholderType.JSON,
description="Training flavor")
    )
)# Training flavors
)

# Define a model registration phase using ModelStep. The output of JobStep is used as the input of
ModelStep.

# Define model name parameters.
model_name = wf.Placeholder(name="placeholder_name", placeholder_type=wf.PlaceholderType.STR)

model_registration = wf.steps.ModelStep(
    name="model_registration", # Name of the model registration phase. The name contains a maximum of
64 characters, including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and
must be unique in a workflow.
    title="Model Registration", # Title
    inputs=wf.steps.ModelInput(name='model_input', data=job_step.outputs["train_url"].as_input()), # The
output of JobStep is used as the input of ModelStep.

outputs=wf.steps.ModelOutput(name='model_output',model_config=wf.steps.ModelConfig(model_name=mo
del_name, model_type="TensorFlow")), # ModelStep outputs
    depend_steps=job_step # Preceding job phase
)
# job_step is an instance object of wf.steps.JobStep and train_url is the value of the name field of
wf.steps.JobOutput.

workflow = wf.Workflow(
    name="model-step-demo",
    desc="this is a demo workflow",
    steps=[job_step, model_registration],
    storages=[storage]
)
```

Registering a Model from a Training Job (Model Source: A Trained Model Stored in OBS)

```
import modelarts.workflow as wf
# Define a model registration phase using ModelStep. The input is from OBS.

# Define the OBS data.
obs = wf.data.OBSPlaceholder(name = "obs_placeholder_name", object_type = "directory" ) # object_type
must be file or directory.

# Define model name parameters.
model_name = wf.Placeholder(name="placeholder_name", placeholder_type=wf.PlaceholderType.STR)

model_registration = wf.steps.ModelStep(
    name="model_registration", # Name of the model registration phase. The name contains a maximum of
64 characters, including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and
must be unique in a workflow.
    title="Model Registration", # Title
    inputs=wf.steps.ModelInput(name='model_input', data=obs), # ModelStep inputs are configured when
the workflow is running. You can also use wf.data.OBSPath(obs_path="fake_obs_path") for the data field.

outputs=wf.steps.ModelOutput(name='model_output',model_config=wf.steps.ModelConfig(model_name=mo
del_name, model_type="TensorFlow"))# ModelStep outputs
)

workflow = wf.Workflow(
    name="model-step-demo",
```

```
desc="this is a demo workflow",
steps=[model_registration]
)
```

Registering a Model Using a Template

```
import modelarts.workflow as wf
# Define a model registration phase using ModelStep. Register a model using a preset template.

# Define a preset template object. Fields in the template object can be represented by placeholders.
template = wf.steps.Template(
    template_id="fake_template_id",
    infer_format="fake_infer_format",
    template_inputs=[
        wf.steps.TemplateInputs(
            input_id="fake_input_id",
            input="fake_input_file"
        )
    ]
)

# Define model name parameters.
model_name = wf.Placeholder(name="placeholder_name", placeholder_type=wf.PlaceholderType.STR)

model_registration = wf.steps.ModelStep(
    name="model_registration", # Name of the model registration phase. The name contains a maximum of
    # 64 characters, including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and
    # must be unique in a workflow.
    title="Model Registration", # Title
    outputs=wf.steps.ModelOutput(
        name='model_output',
        model_config=wf.steps.ModelConfig(
            model_name=model_name,
            model_type="Template",
            template=template
        )
    ) # ModelStep outputs
)

workflow = wf.Workflow(
    name="model-step-demo",
    desc="this is a demo workflow",
    steps=[model_registration]
)
```

Registering a Model from a Custom Image

```
import modelarts.workflow as wf
# Define a model registration phase using ModelStep. The input is from the URL of a custom image.

# Define the image data.
swr = wf.data.SWRImagePlaceholder(name="placeholder_name")

# Define model name parameters.
model_name = wf.Placeholder(name="placeholder_name", placeholder_type=wf.PlaceholderType.STR)

model_registration = wf.steps.ModelStep(
    name="model_registration", # Name of the model registration phase. The name contains a maximum of
    # 64 characters, including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and
    # must be unique in a workflow.
    title="Model Registration", # Title
    inputs=wf.steps.ModelInput(name="input",data=swr), # ModelStep inputs are configured when the
    # workflow is running. You can also use wf.data.SWRImage(swr_path="fake_path") for the data field.

    outputs=wf.steps.ModelOutput(name='model_output',model_config=wf.steps.ModelConfig(model_name=model_name,model_type="TensorFlow"))# ModelStep outputs
)

workflow = wf.Workflow(
```

```
name="model-step-demo",
desc="this is a demo workflow",
steps=[model_registration]
)
```

Registering a Model Using a Custom Image and OBS

```
import modelarts.workflow as wf
# Define a model registration phase using ModelStep. The input is from the URL of a custom image.

# Define the image data.
swr = wf.data.SWRImagePlaceholder(name="placeholder_name")

# Define OBS model data.
model_obs = wf.data.OBSPlaceholder(name = "obs_placeholder_name", object_type = "directory" ) #
object_type must be file or directory.

# Define model name parameters.
model_name = wf.Placeholder(name="placeholder_name", placeholder_type=wf.PlaceholderType.STR)

model_registration = wf.steps.ModelStep(
    name="model_registration", # Name of the model registration phase. The name contains a maximum of
    64 characters, including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and
    must be unique in a workflow.
    title="Model Registration", # Title
    inputs=[
        wf.steps.ModelInput(name="input",data=swr), # ModelStep inputs are configured when the workflow
        is running. You can also use wf.data.SWRImage(swr_path="fake_path") for the data field.
        wf.steps.ModelInput(name="input",data=model_obs) # ModelStep inputs are configured when the
        workflow is running. You can also use wf.data.OBSPath(obs_path="fake_obs_path") for the data field.
    ],
    outputs=wf.steps.ModelOutput(
        name='model_output',
        model_config=wf.steps.ModelConfig(
            model_name=model_name,
            model_type="Custom",
            dynamic_load_mode="Single"
        )
    ) # ModelStep outputs
)

workflow = wf.Workflow(
    name="model-step-demo",
    desc="this is a demo workflow",
    steps=[model_registration]
)
```

Registering a Model Using a Subscribed Model and OBS

This mode is similar to the custom image + OBS mode, except that you obtain a custom image from a subscribed model.

Example:

```
import modelarts.workflow as wf

# Define the subscribed model object.
base_model = wf.data.GalleryModel(subscription_id="fake_subscription_id", version_num="fake_version") #
Model subscribed to from AI Gallery, generally published by a developer

# Define OBS model data.
model_obs = wf.data.OBSPlaceholder(name = "obs_placeholder_name", object_type = "directory" ) #
object_type must be file or directory.

# Define model name parameters.
model_name = wf.Placeholder(name="placeholder_name", placeholder_type=wf.PlaceholderType.STR)

model_registration = wf.steps.ModelStep(
```



```
name="model_registration", # Name of the model registration phase. The name contains a maximum of
64 characters, including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and
must be unique in a workflow.
title="Model Registration", # Title
inputs=[
    wf.steps.ModelInput(name="input",data=base_model) # Use a subscribed model as the ModelStep
input.
    wf.steps.ModelInput(name="input",data=model_obs) # ModelStep inputs are configured when the
workflow is running. You can also use wf.data.OBSPath(obs_path="fake_obs_path") for the data field.
],
outputs=wf.steps.ModelOutput(
    name='model_output',
    model_config=wf.steps.ModelConfig(
        model_name=model_name,
        model_type="Custom",
        dynamic_load_mode="Single"
    )
) # ModelStep outputs
)

workflow = wf.Workflow(
    name="model-step-demo",
    desc="this is a demo workflow",
    steps=[model_registration]
)
```

In the preceding example, the system automatically obtains the custom image from the subscribed model and registers and generates a model based on the entered OBS model path. **model_obs** can be replaced with the dynamic output of JobStep.

NOTE

The value of **model_type** can be **TensorFlow**, **MXNet**, **Caffe**, **Spark_MLLib**, **Scikit_Learn**, **XGBoost**, **Image**, **PyTorch**, **Template**, or **Custom**.

If **model_type** is not set for **wf.steps.ModelConfig**, **TensorFlow** is used by default.

- If the model type of your workflow does not need to be changed, refer to the preceding examples.
- If the model type of your workflow needs to be changed in multiple executions, write the parameter using placeholders.

```
model_type = wf.Placeholder(name="placeholder_name",
placeholder_type=wf.PlaceholderType.ENUM, default="TensorFlow",
enum_list=["TensorFlow", "MXNet", "Caffe", "Spark_MLLib", "Scikit_Learn",
"XGBoost", "Image", "PyTorch", "Template", "Custom"], description="Model
type")
```

4.3.4.7 Creating a Service Deployment Phase

Description

This phase integrates capabilities of ModelArts service management to enable service deployment and update in a workflow. The application scenarios are as follows:

- Deploying a model as a web service
- Updating an existing service (gray update supported)

Parameter Overview

You can use ServiceStep to create a service deployment phase. The following is an example of defining a ServiceStep.

Table 4-60 ServiceStep

Parameter	Description	Mandatory	Data Type
name	Name of a service deployment phase. The name contains a maximum of 64 characters, including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and must be unique in a workflow.	Yes	str
inputs	Inputs of the service deployment phase.	No	ServiceInput or ServiceInput list
outputs	Outputs of the service deployment phase.	Yes	ServiceOutput or ServiceOutput list
title	Title for frontend display.	No	str
description	Description of the service deployment phase.	No	str
policy	Phase execution policy.	No	StepPolicy
depend_steps	Dependent phases.	No	Step or step list

Table 4-61 ServiceInput

Parameter	Description	Mandatory	Data Type
name	Input name of the service deployment phase. The name can contain a maximum of 64 characters, including only letters, digits, underscores (_), and hyphens (-), and must start with a letter. The input name of a step must be unique.	Yes	str

Parameter	Description	Mandatory	Data Type
data	Input data object of the service deployment phase.	Yes	Model list or service object. Currently, only ServiceInputPlaceholder, ServiceData, and ServiceUpdatePlaceholder are supported.

Table 4-62 ServiceOutput

Parameter	Description	Mandatory	Data Type
name	Output name of the service deployment phase. The name can contain a maximum of 64 characters, including only letters, digits, underscores (_), and hyphens (-), and must start with a letter. The output name of a step must be unique.	Yes	str
service_config	Configurations for service deployment.	Yes	ServiceConfig

Table 4 ServiceConfig

Parameter	Description	Mandatory	Data Type
infer_type	Inference mode. The value can be real-time, batch, or edge. The default value is real-time. • real-time: real-time service. The model is deployed as a web service. • batch: batch service. A batch service can perform inference on batch data and automatically stops after data processing is completed. • edge: edge service. A model is deployed as a web service on an edge node through IEF. You must create an edge node on IEF beforehand.	Yes	str
service_name	Service name. Enter 1 to 64 characters. Only letters, digits, hyphens (-), and underscores (_) are allowed. NOTE If you do not specify this parameter, the default service name is generated automatically.	No	str, Placeholder
description	Service description, which contains a maximum of 100 characters. By default, this parameter is left blank.	No	str
vpc_id	ID of the VPC to which a real-time service instance is deployed. By default, this parameter is left blank. In this case, ModelArts allocates a dedicated VPC to each user, and users are isolated from each other. To access other service components in the VPC of the service instance, set this parameter to the ID of the corresponding VPC. Once a VPC is configured, it cannot be modified. If both vpc_id and cluster_id are configured, only the dedicated resource pool takes effect.	No	str

Parameter	Description	Mandatory	Data Type
subnet_network_id	ID of a subnet. By default, this parameter is left blank. This parameter is mandatory when vpc_id is configured. Enter the network ID displayed in the subnet details on the VPC management console. A subnet provides dedicated network resources that are isolated from other networks.	No	str
security_group_id	Security group. By default, this parameter is left blank. This parameter is mandatory when vpc_id is configured. A security group is a virtual firewall that provides secure network access control policies for service instances. A security group must contain at least one inbound rule to permit the requests whose protocol is TCP, source address is 0.0.0.0/0 , and port number is 8080 .	No	str
cluster_id	ID of a dedicated resource pool. By default, this parameter is left blank, indicating that no dedicated resource pool is used. When using a dedicated resource pool to deploy services, ensure that the cluster is running properly. After this parameter is configured, the network configuration of the cluster is used, and the vpc_id parameter does not take effect. If both this parameter and cluster_id in real-time config are configured, cluster_id in real-time config is preferentially used.	No	str
additional_properties	Additional configurations.	No	dict
apps	Whether to enable application authentication for service deployment. Multiple application names can be entered.	No	str, Placeholder, list
envs	Environment variables.	No	dict

Example:

```
example = ServiceConfig()  
# This object is used in the output of the service deployment phase.
```

If there is no special requirement, use the default values.

Examples

There are three scenarios:

- Deploying a real-time service
- Modifying a real-time service
- Getting the inference address from the service deployment phase

Deploying a Real-Time Service

```
import modelarts.workflow as wf  
# Use ServiceStep to define a service deployment phase and specify a model for service deployment.  
  
# Define model name parameters.  
model_name = wf.Placeholder(name="placeholder_name", placeholder_type=wf.PlaceholderType.STR)  
  
service_step = wf.steps.ServiceStep(  
    name="service_step", # Name of the service deployment phase. The name contains a maximum of 64  
                        # characters, including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and  
                        # must be unique in a workflow.  
    title="Deploying a Service", # Title  
    inputs=wf.steps.ServiceInput(name="si_service_ph",  
                                data=wf.data.ServiceInputPlaceholder(name="si_placeholder1",  
                                                                    # Restrictions on the model name: Only the model  
                                                                    name specified here can be used in the running state; use the same model name as model_name of the  
                                                                    model registration phase.  
                                                                    model_name=model_name)), # ServiceStep inputs  
    outputs=wf.steps.ServiceOutput(name="service_output") # ServiceStep outputs  
)  
  
workflow = wf.Workflow(  
    name="service-step-demo",  
    desc="this is a demo workflow",  
    steps=[service_step]  
)
```

Modifying a Real-Time Service

Scenario: When you use a new model version to update an existing service, ensure that the name of the new model version is the same as that of the deployed service.

```
import modelarts.workflow as wf  
# Use ServiceStep to define a service deployment phase and specify a model for service update.  
  
# Define model name parameters.  
model_name = wf.Placeholder(name="placeholder_name", placeholder_type=wf.PlaceholderType.STR)  
  
# Define a service object.  
service = wf.data.ServiceUpdatePlaceholder(name="placeholder_name")  
  
service_step = wf.steps.ServiceStep(  
    name="service_step", # Name of the service deployment phase. The name contains a maximum of 64  
                        # characters, including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and  
                        # must be unique in a workflow.  
    title="Service Update", # Title  
    inputs=[wf.steps.ServiceInput(name="si2",  
                                data=wf.data.ServiceInputPlaceholder(name="si_placeholder2",  
                                                                    # Restrictions on the model name: Only the model
```

```
name specified here can be used in the running state.
        model_name=model_name)),
    wf.steps.ServiceInput(name="si_service_data", data=service) # ServiceStep inputs are configured
when the workflow is running. You can also use wf.data.ServiceData(service_id="fake_service") for the
data field.
], # ServiceStep inputs
outputs=wf.steps.ServiceOutput(name="service_output") # ServiceStep outputs
)

workflow = wf.Workflow(
    name="service-step-demo",
    desc="this is a demo workflow",
    steps=[service_step]
)
```

Getting the Inference Address from the Service Deployment Phase

The service deployment phase supports the output of the inference address. You can use the **get_output_variable("access_address")** method to obtain the output and use it in subsequent phases.

- For services deployed in the public resource pool, you can use **access_address** to obtain the inference address registered on the public network from the output.
- For services deployed in a dedicated resource pool, you can get the internal inference address from the output using **cluster_inner_access_address**, in addition to the public inference address. The internal address can only be accessed by other inference services.

```
import modelarts.workflow as wf

# Define model name parameters.
sub_model_name = wf.Placeholder(name="si_placeholder1",
placeholder_type=wf.PlaceholderType.STR)

sub_service_step = wf.steps.ServiceStep(
    name="sub_service_step", # Name of the service deployment phase. The name contains a
maximum of 64 characters, including only letters, digits, underscores (_), and hyphens (-). It must
start with a letter and must be unique in a workflow.
    title="Subservice", # Title
    inputs=wf.steps.ServiceInput(
        name="si_service_ph",
        data=wf.data.ServiceInputPlaceholder(name="si_placeholder1", model_name=sub_model_name)
    ),# ServiceStep inputs
    outputs=wf.steps.ServiceOutput(name="service_output") # ServiceStep outputs
)

main_model_name = wf.Placeholder(name="si_placeholder2",
placeholder_type=wf.PlaceholderType.STR)

# Obtain the inference address output by the subservice and transfer the address to the main service
through envs.
main_service_config = wf.steps.ServiceConfig(
    infer_type="real-time",
    envs={"infer_address":
sub_service_step.outputs["service_output"].get_output_variable("access_address")} # Obtain the
inference address output by the subservice and transfer the address to the main service through envs.
)

main_service_step = wf.steps.ServiceStep(
    name="main_service_step", # Name of the service deployment phase. The name contains a
maximum of 64 characters, including only letters, digits, underscores (_), and hyphens (-). It must
start with a letter and must be unique in a workflow.
    title="Main service", # Title
    inputs=wf.steps.ServiceInput(
        name="si_service_ph",
```

```
data=wf.data.ServiceInputPlaceholder(name="si_placeholder2",
model_name=main_model_name)
),# ServiceStep inputs
outputs=wf.steps.ServiceOutput(name="service_output", service_config=main_service_config), #
ServiceStep outputs
depend_steps=sub_service_step
)

workflow = wf.Workflow(
name="service-step-demo",
desc="this is a demo workflow",
steps=[sub_service_step, main_service_step]
)
```

Configuring Information for Deploying a Synchronous Service

After the service deployment phase is started in the development state (usually a notebook instance), configure the information based on the following format in the logs.

Please enter the ServiceInputPlaceholder "service_model" in the following format:

```
"[[{'model_name': '*****', 'model_version': 'v1', 'specification': '*****', 'weight': 50}, {'model_name': '*****', 'model_version': 'v2', 'specification': '*****', 'weight': 30}, {'model_name': '*****', 'model_version': 'v3', 'specification': '*****', 'weight': 20}], [{'k1': 'v1', 'k2': 'v2'}], {'model_name': '*****', 'model_version': 'v1', 'specification': '*****', 'weight': 20, 'envs': {'*****': '*****', '*****': '*****'}}]"
```

Note that: (1) The "[" at the beginning and "]" at the end are required.
(2) The sum of the weights must be equal to 100.
(3) All model must have the same model name. Two model versions cannot be the same.

1. On the ModelArts console, choose **Development Workspace > Workflow** from the navigation pane.
2. Configure the information after the service deployment phase is started. After the configuration, click **Next**.

If you want to start the deployment phase automatically without manual configuration, configure the **ServiceInputConfig** or **ServiceConfig** parameter in the code in advance, for example, **service_config=wf.steps.ServiceConfig(cluster_id="XX")**. If all required parameters are specified, the service deployment phase automatically starts.

Configurations

Attribute

Status ● Awaiting Input

Started At Oct 10, 2024 14:25:49 GMT+08:00

Duration 00:00:05

Updated At Oct 10, 2024 14:25:54 GMT+08:00

Input

★ Real-Time S... ☐

Model Source **My Model** Workflow step

Model and Version workflow_auto_test_model 0.0.1/Not... Q

Resource Pool **Public Resource Pool** Dedicated Resource Pool

Specifications CPU 2 core 8GB

Price ¥ 0.80 hour

Traffic Ratio (%) - 100 +

Compute Nodes - 1 +

Environment Variable ☐ Add Environment Variable

Parameters

★ Auto Stop ☒

Enable this option to automatically stop the real-time service at the time you specify. You will not be billed for the service after the service is stopped.

☐ 1 hour later ☒ 2 hours later ☐ 4 hours later ☐ 6 hours later ☐ Custom

Next

Configuring Information for Deploying an Asynchronous Service

1. On the ModelArts console, choose **Workflow** from the navigation pane.
2. Configure the information after the service deployment phase is started. Select an asynchronous inference model and a version, and configure service startup parameters. After the configuration, click **Next**.

 NOTE

After you select the required model and version, the system automatically matches the service startup parameters.

4.3.5 Creating a Multi-Branch Workflow

4.3.5.1 Multi-Branch Workflow

You can implement multi-branch in two ways. However, the condition phase is limited to two branches. [Configuring Phase Parameters to Control Branch Execution](#) allows you to replace the ConditionStep capability without adding new phases, offering more flexibility.

[Creating a Condition Phase to Control Branch Execution](#) is used for conditional branching in the execution of phases based on condition value comparison or metrics output by the preceding phase.

[Configuring Phase Parameters to Control Branch Execution](#) is used for complex scenarios that involve multiple branches. When each execution starts, the workflow decides which branches to run and which ones to skip based on the relevant configuration information. This way, only some branches are executed. This function has a similar use case as ConditionStep, but it is more powerful.

4.3.5.2 Creating a Condition Phase to Control Branch Execution

Description

This phase is used for conditional branching in the execution of phases based on condition value comparison or metrics output by the preceding phase. The application scenarios are as follows:

You need to determine the subsequent process based on different input values. If you need to determine whether to retrain or register a model based on the model precision output by the training phase, you can use this phase to control the process.

Parameter Overview

You can use ConditionStep to create a condition phase. The following is an example of defining a ConditionStep.

Table 4-63 ConditionStep

Parameter	Description	Mandatory	Data Type
name	Name of a condition phase. The name contains a maximum of 64 characters, including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and must be unique in a workflow.	Yes	str
conditions	List of conditions. The AND operation is used for multiple conditions.	Yes	Condition or condition list
if_then_steps	Steps to be executed if the calculation result of the condition expression is True .	No	str or str list
else_then_steps	Steps to be executed if the calculation result of the condition expression is False .	No	str or str list
title	Title for frontend-phase display.	No	str
description	Description of a condition phase.	No	str
depend_steps	Dependent phases.	No	Step or step list

Table 4-64 Condition

Parameter	Description	Mandatory	Data Type
condition_type	Condition type. The "=", ">", ">=", "in", "<", "<=", "!=", and "or" operators are supported.	Yes	ConditionTypeEnum
left	Left value of a condition expression.	Yes	int, float, str, bool, Placeholder, Sequence, Condition, MetricInfo

Parameter	Description	Mandator y	Data Type
right	Right value of a condition expression	Yes	int, float, str, bool, Placeholder, Sequence, Condition, MetricInfo

Table 4-65 MetricInfo

Parameter	Description	Mandator y	Data Type
input_data	Metric input. Currently, only the output of JobStep is supported.	Yes	JobStep output
json_key	Key value corresponding to the metric information to be obtained	Yes	str

Description of the structure:

- **Condition object, which consists of the condition type, left value, and right value**
 - The condition type is obtained from ConditionTypeEnum. The "==", ">", ">=", "in", "<", "<=", "!=", and "or" operators are supported. The following table describes the mapping.

Enumeration	Operator
ConditionTypeEnum.EQ	==
ConditionTypeEnum.GT	>
ConditionTypeEnum.GTE	>=
ConditionTypeEnum.IN	in
ConditionTypeEnum.LT	<
ConditionTypeEnum.LTE	<=
ConditionTypeEnum.NOT	!=
ConditionTypeEnum.OR	or

- The left and right values support the following types: integer, float, string, bool, placeholder, sequence, condition, and MetricInfo.
- A condition phase supports a list of condition objects. The && operation is performed between multiple conditions.

- **if_then_steps and else_then_steps**
 - **if_then_steps** indicates a list of phases that are ready for execution if conditions evaluate to **true**. In this case, steps in **else_then_steps** are skipped.
 - **else_then_steps** indicates a list of phases that are ready for execution if conditions evaluate to **false**. In this case, steps in **if_then_steps** are skipped.

Examples

Refer to simple or advanced examples as needed.

Simple Examples

- Implemented using parameter configurations

```
import modelarts.workflow as wf

left_value = wf.Placeholder(name="left_value", placeholder_type=wf.PlaceholderType.BOOL,
                             default=True)

# Condition object
condition = wf.steps.Condition(condition_type=wf.steps.ConditionTypeEnum.EQ, left=left_value,
                                right=True) # Condition object, including the type, left value, and right value.

# Condition phase
condition_step = wf.steps.ConditionStep(
    name="condition_step_test", # Name of the condition phase. The name contains a maximum of 64
                                # characters, including only letters, digits, underscores (_), and hyphens (-). It must start with a letter
                                # and must be unique in a workflow.
    conditions=condition, # Condition objects. The relationship between the conditions is &&.
    if_then_steps="job_step_1", # If conditions evaluate to true, job_step_1 is ready for execution, and
                                # job_step_2 is skipped.
    else_then_steps="job_step_2" # If conditions evaluate to false, job_step_2 is ready for execution,
                                # and job_step_1 is skipped.
)

# This phase is used only as an example. You need to supplement other fields as required.
job_step_1 = wf.steps.JobStep(
    name="job_step_1",
    depend_steps=condition_step
)

# This phase is used only as an example. You need to supplement other fields as required.
model_step_1 = wf.steps.ModelStep(
    name="model_step_1",
    depend_steps=job_step_1
)

# This phase is used only as an example. You need to supplement other fields as required.
job_step_2 = wf.steps.JobStep(
    name="job_step_2",
    depend_steps=condition_step
)

# This phase is used only as an example. You need to supplement other fields as required.
model_step_2 = wf.steps.ModelStep(
    name="model_step_2",
    depend_steps=job_step_2
)

workflow = wf.Workflow(
    name="condition-demo",
    desc="this is a demo workflow",
```

```
steps=[condition_step, job_step_1, job_step_2, model_step_1, model_step_2]
)
```

NOTE

Scenario description: **job_step_1** and **job_step_2** indicate two training phases that depend on **condition_step**. **condition_step** parameters determine the subsequent phase execution.

Execution analysis:

- If the default value of **left_value** is **True**, the calculation result of the condition logical expression is **True**. Then, **job_step_1** is executed, **job_step_2** is skipped, and all phases contained in the branches that use **job_step_2** as the unique root node are skipped. That is, **model_step_2** is skipped. Therefore, **condition_step**, **job_step_1**, and **model_step_1** are executed.
- If **left_value** is set to **False**, the calculation result of the condition logical expression is **False**. Then, **job_step_2** is executed, **job_step_1** is skipped, and all phases contained in the branches that use **job_step_1** as the unique root node are skipped. That is, **model_step_1** is skipped, and **condition_step**, **job_step_2**, and **model_step_2** are executed.
- Implemented by obtaining the metric information output by JobStep

from modelarts import workflow as wf

```
# Create an OutputStorage object to centrally manage training output directories.
storage = wf.data.Storage(name="storage_name", title="title_info", with_execution_id=True,
create_dir=True, description="description_info") # The name field is mandatory, and the title and
description fields are optional.

# Define the input OBS object.
obs_data = wf.data.OBSPlaceholder(name="obs_placeholder_name", object_type="directory")

# Use JobStep to define a training phase, and use OBS to store the output.
job_step = wf.steps.JobStep(
    name="training_job", # Name of a training phase. The name contains a maximum of 64
characters, including only letters, digits, underscores (_), and hyphens (-). It must start with a letter
and must be unique in a workflow.
    title="Image classification training", # Title, which defaults to the value of name.
    algorithm=wf.AIGalleryAlgorithm(
        subscription_id="subscription_id", # Subscription ID of the subscribed algorithm
        item_version_id="item_version_id", # Algorithm version ID. You can also enter the version
number instead.
        parameters=[]
    ), # Algorithm used for training. An algorithm subscribed to in AI Gallery is used in this example. If
the value of an algorithm hyperparameter does not need to be changed, you do not need to
configure the hyperparameter in parameters. Hyperparameter values will be automatically filled.
    inputs=wf.steps.JobInput(name="data_url", data=obs_data),
    outputs=[

wf.steps.JobOutput(name="train_url", obs_config=wf.data.OBSOutputConfig(obs_path=storage.join("dir
ectory_path"))),
    wf.steps.JobOutput(name="metrics",
metrics_config=wf.data.MetricsConfig(metric_files=storage.join("directory_path/metrics.json",
create_dir=False))) # Metric output path. Metric information is automatically output by the job script
based on the specified data format. (In the example, the metric information needs to be output to the
metrics.json file in the training output directory.)
    ],
    spec=wf.steps.JobSpec(
        resource=wf.steps.JobResource(
            flavor=wf.Placeholder(name="train_flavor", placeholder_type=wf.PlaceholderType.JSON,
description="Training flavor")
        )
    ) # Training flavors
)
```

```
# Define a condition object.
condition_lt = wf.steps.Condition(
    condition_type=wf.steps.ConditionTypeEnum.LT,
    left=wf.steps.MetricInfo(job_step.outputs["metrics"].as_input(), "accuracy"),
    right=0.5
)

condition_step = wf.steps.ConditionStep(
    name="condition_step_test", # Name of the condition phase. The name contains a maximum of 64
    characters, including only letters, digits, underscores (_), and hyphens (-). It must start with a letter
    and must be unique in a workflow.
    conditions=condition_lt, # Condition objects. The relationship between the conditions is &&.
    if_then_steps="training_job_retrain", # If conditions evaluate to true, training_job_retrain is ready
    for execution, and model_registration is skipped.
    else_then_steps="model_registration", # If conditions evaluate to false, model_registration is
    ready for execution, and training_job_retrain is skipped.
    depend_steps=job_step
)

# Use JobStep to define a training phase, and use OBS to store the output.
job_step_retrain = wf.steps.JobStep(
    name="training_job_retrain", # Name of a training phase. The name contains a maximum of 64
    characters, including only letters, digits, underscores (_), and hyphens (-). It must start with a letter
    and must be unique in a workflow.
    title="Image classification retraining", # Title, which defaults to the value of name.
    algorithm=wf.AIGalleryAlgorithm(
        subscription_id="subscription_id", # Subscription ID of the subscribed algorithm
        item_version_id="item_version_id", # Algorithm version ID. You can also enter the version
        number instead.
        parameters=[]
    ), # Algorithm used for training. An algorithm subscribed to in AI Gallery is used in this example. If
    the value of an algorithm hyperparameter does not need to be changed, you do not need to
    configure the hyperparameter in parameters. Hyperparameter values will be automatically filled.
    inputs=wf.steps.JobInput(name="data_url", data=obs_data),
    outputs=[

wf.steps.JobOutput(name="train_url",obs_config=wf.data.OBSOutputConfig(obs_path=storage.join("dir
ectory_path_retrain"))),
        wf.steps.JobOutput(name="metrics",
        metrics_config=wf.data.MetricsConfig(metric_files=storage.join("directory_path_retrain/metrics.json",
        create_dir=False))) # Metric output path. Metric information is automatically output by the job script
        based on the specified data format. (In the example, the metric information needs to be output to the
metrics.json file in the training output directory.)
    ],
    spec=wf.steps.JobSpec(
        resource=wf.steps.JobResource(
            flavor=wf.Placeholder(name="train_flavor_retrain",
            placeholder_type=wf.PlaceholderType.JSON, description="Training flavor")
        )
    ), # Training flavors
    depend_steps=condition_step
)

# Define model name parameters.
model_name = wf.Placeholder(name="placeholder_name", placeholder_type=wf.PlaceholderType.STR)

model_step = wf.steps.ModelStep(
    name="model_registration", # Name of the model registration phase. The name contains a
    maximum of 64 characters, including only letters, digits, underscores (_), and hyphens (-). It must
    start with a letter and must be unique in a workflow.
    title="Model Registration", # Title
    inputs=wf.steps.ModelInput(name='model_input', data=job_step.outputs["train_url"].as_input()), #
    job_step output is used as the input.
    outputs=wf.steps.ModelOutput(name='model_output',
    model_config=wf.steps.ModelConfig(model_name=model_name, model_type="TensorFlow")), #
    ModelStep outputs
    depend_steps=condition_step,
```

```
)  
  
workflow = wf.Workflow(  
    name="condition-demo",  
    desc="this is a demo workflow",  
    steps=[job_step, condition_step, job_step_retrain, model_step],  
    storages=storage  
)
```

In this example, ConditionStep obtains the accuracy output by **job_step** and compares it with the preset value to determine whether to retrain or register the model. When the accuracy output by **job_step** is less than the threshold 0.5, the calculation result of **condition_It** is **True**. In this case, **job_step_retrain** runs and **model_step** skips. Otherwise, **job_step_retrain** skips and **model_step** runs.

NOTE

For details about the format requirements of the metric file generated by **job_step**, see [Creating a Training Job Phase](#). In the condition phase, only the metric data whose type is float can be used as the input.

The following is an example of the **metrics.json** file:

```
[  
  {  
    "key": "loss",  
    "title": "loss",  
    "type": "float",  
    "data": {  
      "value": 1.2  
    }  
  },  
  {  
    "key": "accuracy",  
    "title": "accuracy",  
    "type": "float",  
    "data": {  
      "value": 0.8  
    }  
  }  
]
```

Advanced Examples

```
import modelarts.workflow as wf  
  
left_value = wf.Placeholder(name="left_value", placeholder_type=wf.PlaceholderType.BOOL, default=True)  
condition1 = wf.steps.Condition(condition_type=wf.steps.ConditionTypeEnum.EQ, left=left_value, right=True)  
  
internal_condition_1 = wf.steps.Condition(condition_type=wf.steps.ConditionTypeEnum.GT, left=10, right=9)  
internal_condition_2 = wf.steps.Condition(condition_type=wf.steps.ConditionTypeEnum.LT, left=10, right=9)  
  
# The result of condition2 is internal_condition_1 || internal_condition_2.  
condition2 = wf.steps.Condition(condition_type=wf.steps.ConditionTypeEnum.OR, left=internal_condition_1,  
    right=internal_condition_2)  
  
condition_step = wf.steps.ConditionStep(  
    name="condition_step_test", # Name of the condition phase. The name contains a maximum of 64  
    characters, including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and  
    must be unique in a workflow.  
    conditions=[condition1, condition2], # Condition objects. The relationship between the conditions is &&.  
    if_then_steps=["job_step_1"], # If conditions evaluate to true, job_step_1 is ready for execution, and  
    job_step_2 is skipped.  
    else_then_steps=["job_step_2"] # If conditions evaluate to false, job_step_2 is ready for execution, and  
    job_step_1 is skipped.  
)
```

```
# This phase is used only as an example. You need to supplement other fields as required.
job_step_1 = wf.steps.JobStep(
    name="job_step_1",
    depend_steps=condition_step
)

# This phase is used only as an example. You need to supplement other fields as required.
job_step_2 = wf.steps.JobStep(
    name="job_step_2",
    depend_steps=condition_step
)

workflow = wf.Workflow(
    name="condition-demo",
    desc="this is a demo workflow",
    steps=[condition_step, job_step_1, job_step_2],
)
```

ConditionStep supports nested condition phases. You can flexibly design tit based on different scenarios.

NOTE

The condition phase can only support two branches, which is very limiting. You can use the new branch function to replace the ConditionStep capability without creating new phases. For details, see [Configuring Phase Parameters to Control Branch Execution](#).

4.3.5.3 Configuring Phase Parameters to Control Branch Execution

Function

You can use parameters or metrics from training output to decide whether to run a phase. This way, you can control the process.

Application Scenarios

This function is used for complex scenarios that involve multiple branches. When each execution starts, the workflow decides which branches to run and which ones to skip based on the relevant configuration information. This way, only some branches are executed. This function has a similar use case as ConditionStep, but it is more powerful. This function applies to the dataset creation phase, labeling phase, dataset import phase, dataset release phase, job phase, model registration phase, and service deployment phase.

Controlling the Execution of a Single Phase

- Implemented using parameter configurations

```
from modelarts import workflow as wf

condition_equal = wf.steps.Condition(condition_type=wf.steps.ConditionTypeEnum.EQ,
left=wf.Placeholder(name="is_skip", placeholder_type=wf.PlaceholderType.BOOL), right=True)

# Create an OutputStorage object to centrally manage training output directories.
storage = wf.data.OutputStorage(name="storage_name", title="title_info",
description="description_info") # Only name is mandatory.

# Define the input OBS object.
obs_data = wf.data.OBSPlaceholder(name="obs_placeholder_name", object_type="directory")
```



```
# Use JobStep to define a training phase, and use OBS to store the output.
job_step = wf.steps.JobStep(
    name="training_job", # Name of a training phase. The name contains a maximum of 64
    characters, including only letters, digits, underscores (_), and hyphens (-). It must start with a letter
    and must be unique in a workflow.
    title="Image classification training", # Title, which defaults to the value of name.
    algorithm=wf.AIGalleryAlgorithm(
        subscription_id="subscription_id", # Subscription ID of the subscribed algorithm
        item_version_id="item_version_id", # Algorithm version ID. You can also enter the version
        number instead.
        parameters=[]

    ), # Algorithm used for training. An algorithm subscribed to in AI Gallery is used in this example. If
    the value of an algorithm hyperparameter does not need to be changed, you do not need to
    configure the hyperparameter in parameters. Hyperparameter values will be automatically filled.

    inputs=wf.steps.JobInput(name="data_url", data=obs_data),
    # JobStep input is configured when the workflow is running. You can also use
    data=wf.data.OBSPath(obs_path="fake_obs_path") for the data field.
    outputs=wf.steps.JobOutput(name="train_url",
        obs_config=wf.data.OBSOutputConfig(obs_path=storage.join("directory_path"))),
    # JobStep output
    spec=wf.steps.JobSpec(
        resource=wf.steps.JobResource(
            flavor=wf.Placeholder(name="train_flavor", placeholder_type=wf.PlaceholderType.JSON,
            description="Training flavor")

        )
    ), # Training flavors
    policy=wf.steps.StepPolicy(
        skip_conditions=[condition_equal] # Determines whether to skip job_step based on the
        calculation result of skip_conditions.
    )
)

workflow = wf.Workflow(
    name="new-condition-demo",
    desc="this is a demo workflow",
    steps=[job_step],
    storages=storage
)
```

In this example, **job_step** has a skip policy that is controlled by a bool parameter. If the placeholder parameter named **is_skip** is set to **True**, then **job_step** is skipped when **condition_equal** evaluates to **True**. Otherwise, **job_step** is run. For more details about the condition object, see [Creating a Condition Phase to Control Branch Execution](#).

- Implemented by obtaining the metric information output by JobStep from modelarts import workflow as wf

```
# Create an OutputStorage object to centrally manage training output directories.
storage = wf.data.Storage(name="storage_name", title="title_info", with_execution_id=True,
    create_dir=True, description="description_info") # The name field is mandatory, and the title and
description fields are optional.

# Define the input OBS object.
obs_data = wf.data.OBSPlaceholder(name="obs_placeholder_name", object_type="directory")

# Use JobStep to define a training phase, and use OBS to store the output.
job_step = wf.steps.JobStep(
    name="training_job", # Name of a training phase. The name contains a maximum of 64
    characters, including only letters, digits, underscores (_), and hyphens (-). It must start with a letter
    and must be unique in a workflow.
    title="Image classification training", # Title, which defaults to the value of name.
    algorithm=wf.AIGalleryAlgorithm(
        subscription_id="subscription_id", # Subscription ID of the subscribed algorithm
        item_version_id="item_version_id", # Algorithm version ID. You can also enter the version
        number instead.
```

```
parameters=[]

), # Algorithm used for training. An algorithm subscribed to in AI Gallery is used in this example. If
the value of an algorithm hyperparameter does not need to be changed, you do not need to
configure the hyperparameter in parameters. Hyperparameter values will be automatically filled.
inputs=wf.steps.JobInput(name="data_url", data=obs_data),
outputs=[

wf.steps.JobOutput(name="train_url",obs_config=wf.data.OBSOutputConfig(obs_path=storage.join("dir
ectory_path"))),
wf.steps.JobOutput(name="metrics",
metrics_config=wf.data.MetricsConfig(metric_files=storage.join("directory_path/metrics.json",
create_dir=False))) # Metric output path. Metric information is automatically output by the job script
based on the specified data format. (In the example, the metric information needs to be output to the
metrics.json file in the training output directory.)
],
spec=wf.steps.JobSpec(
resource=wf.steps.JobResource(
flavor=wf.Placeholder(name="train_flavor", placeholder_type=wf.PlaceholderType.JSON,
description="Training flavor")
)
) # Training flavors
)

# Define model name parameters.
model_name = wf.Placeholder(name="placeholder_name", placeholder_type=wf.PlaceholderType.STR)

# Define a condition object.
condition_lt = wf.steps.Condition(
condition_type=wf.steps.ConditionTypeEnum.LT,
left=wf.steps.MetricInfo(job_step.outputs["metrics"].as_input(), "accuracy"),
right=0.5
)

model_step = wf.steps.ModelStep(
name="model_registration", # Name of the model registration phase. The name contains a
maximum of 64 characters, including only letters, digits, underscores (_), and hyphens (-). It must
start with a letter and must be unique in a workflow.
title="Model Registration", # Title
inputs=wf.steps.ModelInput(name='model_input', data=job_step.outputs["train_url"].as_input()), #
job_step output is used as the input.
outputs=wf.steps.ModelOutput(name='model_output',
model_config=wf.steps.ModelConfig(model_name=model_name, model_type="TensorFlow")), #
ModelStep outputs
depend_steps=job_step # Preceding job phase
policy=wf.steps.StepPolicy(skip_conditions=condition_lt) # Determines whether to skip model_step
based on the calculation result of skip_conditions.
)

workflow = wf.Workflow(
name="new-condition-demo",
desc="this is a demo workflow",
steps=[job_step, model_step],
storages=storage
)
```

In this example, **model_step** has a skip policy. The model registration depends on whether the accuracy output by **job_step** meets the preset value. When the accuracy output by **job_step** is less than the threshold 0.5, the calculation result of **condition_lt** is **True**. In this case, **model_step** skips. Otherwise, **model_step** runs.

NOTE

For details about the format requirements of the metric file generated by **job_step**, see [Creating a Training Job Phase](#). In the condition phase, only the metric data whose type is float can be used as the input.

The following is an example of the **metrics.json** file:

```
[
  {
    "key": "loss", // Metric data name. The value contains a maximum of 64 characters and cannot
    contain special characters.
    "title": "loss", // Metric data title. The value contains a maximum of 64 characters.
    "type": "float", // Metric data type. Floating point, line chart, histogram, table, and one-
    dimensional table data types are supported.
    "data": {
      "value": 1.2 // Metric data value. For details about the usage examples of different types, see
      Creating a Training Job Phase.
    }
  },
  {
    "key": "accuracy",
    "title": "accuracy",
    "type": "float",
    "data": {
      "value": 0.8
    }
  }
]
```

Controlling Partial Execution of Multiple Branches

```
from modelarts import workflow as wf

# Create an OutputStorage object to centrally manage training output directories.
storage = wf.data.Storage(name="storage_name", title="title_info", with_execution_id=True, create_dir=True,
description="description_info") # The name field is mandatory, and the title and description fields are
optional.

# Define the input OBS object.
obs_data = wf.data.OBSPlaceholder(name="obs_placeholder_name", object_type="directory")

condition_equal_a = wf.steps.Condition(condition_type=wf.steps.ConditionTypeEnum.EQ,
left=wf.Placeholder(name="job_step_a_is_skip", placeholder_type=wf.PlaceholderType.BOOL), right=True)

# Use JobStep to define a training phase, and use OBS to store the output.
job_step_a = wf.steps.JobStep(
    name="training_job_a", # Name of a training phase. The name contains a maximum of 64 characters,
    including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and must be
    unique in a workflow.
    title="Image classification training", # Title, which defaults to the value of name.
    algorithm=wf.AIGalleryAlgorithm(
        subscription_id="subscription_id", # Subscription ID of the subscribed algorithm
        item_version_id="item_version_id", # Algorithm version ID. You can also enter the version number
        instead.
        parameters=[]
    ), # Algorithm used for training. An algorithm subscribed to in AI Gallery is used in this example. If the
    value of an algorithm hyperparameter does not need to be changed, you do not need to configure the
    hyperparameter in parameters. Hyperparameter values will be automatically filled.
    inputs=wf.steps.JobInput(name="data_url", data=obs_data),
    outputs=[wf.steps.JobOutput(name="train_url",
    obs_config=wf.data.OBSOutputConfig(obs_path=storage.join("directory_path_a")))],
    spec=wf.steps.JobSpec(
        resource=wf.steps.JobResource(
            flavor=wf.Placeholder(name="train_flavor", placeholder_type=wf.PlaceholderType.JSON,
            description="Training flavor")
        )
    ), # Training flavors
    policy=wf.steps.StepPolicy(skip_conditions=condition_equal_a)
)

condition_equal_b = wf.steps.Condition(condition_type=wf.steps.ConditionTypeEnum.EQ,
left=wf.Placeholder(name="job_step_b_is_skip", placeholder_type=wf.PlaceholderType.BOOL), right=True)
```

```
# Use JobStep to define a training phase, and use OBS to store the output.
job_step_b = wf.steps.JobStep(
    name="training_job_b", # Name of a training phase. The name contains a maximum of 64 characters,
                           # including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and must be
                           # unique in a workflow.
    title="Image classification training", # Title, which defaults to the value of name.
    algorithm=wf.AIGalleryAlgorithm(
        subscription_id="subscription_id", # Subscription ID of the subscribed algorithm
        item_version_id="item_version_id", # Algorithm version ID. You can also enter the version number
        instead.
        parameters=[]
    ), # Algorithm used for training. An algorithm subscribed to in AI Gallery is used in this example. If the
    # value of an algorithm hyperparameter does not need to be changed, you do not need to configure the
    # hyperparameter in parameters. Hyperparameter values will be automatically filled.
    inputs=wf.steps.JobInput(name="data_url", data=obs_data),
    outputs=[wf.steps.JobOutput(name="train_url",
                                obs_config=wf.data.OBSOutputConfig(obs_path=storage.join("directory_path_b")))],
    spec=wf.steps.JobSpec(
        resource=wf.steps.JobResource(
            flavor=wf.Placeholder(name="train_flavor", placeholder_type=wf.PlaceholderType.JSON,
                                description="Training flavor")
        )
    ), # Training flavors
    policy=wf.steps.StepPolicy(skip_conditions=condition_equal_b)
)

# Define model name parameters.
model_name = wf.Placeholder(name="placeholder_name", placeholder_type=wf.PlaceholderType.STR)

model_step = wf.steps.ModelStep(
    name="model_registration", # Name of the model registration phase. The name contains a maximum of
    # 64 characters, including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and
    # must be unique in a workflow.
    title="Model Registration", # Title
    inputs=wf.steps.ModelInput(name='model_input',
                                data=wf.data.DataConsumptionSelector(data_list=[job_step_a.outputs["train_url"].as_input(),
                                job_step_b.outputs["train_url"].as_input()])), # Select the output of job_step_a or job_step_b as the input.
    outputs=wf.steps.ModelOutput(name='model_output',
                                model_config=wf.steps.ModelConfig(model_name=model_name, model_type="TensorFlow")), # ModelStep
    outputs
    depend_steps=[job_step_a, job_step_b], # Preceding job phase
)

workflow = wf.Workflow(
    name="new-condition-demo",
    desc="this is a demo workflow",
    steps=[job_step_a, job_step_b, model_step],
    storages=storage
)
```

In this example, both **job_step_a** and **job_step_b** have a skip policy that is controlled by parameters. When the parameter values are different, the execution of **model_step** can be divided into the following cases (**model_step** has no skip policy configured, so it follows the default rule).

job_step_a_is_skip	job_step_b_is_skip	Whether to Execute model_step
True	True	No
	False	Yes
False	True	Yes

job_step_a_is_skip	job_step_b_is_skip	Whether to Execute model_step
	False	Yes

 CAUTION

Default rule: A phase is automatically skipped if all the phases it depends on are skipped. Otherwise, the phase is run. This logic can apply to any phase.

Based on the previous example, if you want to override the default rule and make **model_step** run when **job_step_a** and **job_step_b** are skipped, you only need to configure a skip policy in **model_step**. The skip policy takes precedence over the default rule.

4.3.5.4 Configuring Multi-Branch Phase Data

Function

This function is only for the scenario where multiple branches are run. When you create a workflow phase, the data input source of the phase is uncertain. The data input source could be the output of any of the phases it depends on. Only after all dependency phases are run, the valid output is automatically selected as the input based on the actual execution situation.

Examples

```
from modelarts import workflow as wf

condition_equal = wf.steps.Condition(condition_type=wf.steps.ConditionTypeEnum.EQ,
left=wf.Placeholder(name="is_true", placeholder_type=wf.PlaceholderType.BOOL), right=True)
condition_step = wf.steps.ConditionStep(
    name="condition_step",
    conditions=[condition_equal],
    if_then_steps=["training_job_1"],
    else_then_steps=["training_job_2"],
)

# Create an OutputStorage object to centrally manage training output directories.
storage = wf.data.OutputStorage(name="storage_name", title="title_info",
description="description_info") # Only name is mandatory.

# Define the input OBS object.
obs_data = wf.data.OBSPlaceholder(name="obs_placeholder_name", object_type="directory")

# Use JobStep to define a training phase, and use OBS to store the output.
job_step_1 = wf.steps.JobStep(
    name="training_job_1", # Name of a training phase. The name contains a maximum of 64 characters,
including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and must be
unique in a workflow.
    title="Image classification training", # Title, which defaults to the value of name.
    algorithm=wf.AIGalleryAlgorithm(
        subscription_id="subscription_id", # Subscription ID of the subscribed algorithm
        item_version_id="item_version_id", # Algorithm version ID. You can also enter the version number
instead.
        parameters=[]
```

```
), # Algorithm used for training. An algorithm subscribed to in AI Gallery is used in this example. If the
value of an algorithm hyperparameter does not need to be changed, you do not need to configure the
hyperparameter in parameters. Hyperparameter values will be automatically filled.

    inputs=wf.steps.JobInput(name="data_url", data=obs_data),
    # JobStep input is configured when the workflow is running. You can also use
data=wf.data.OBSPath(obs_path="fake_obs_path") for the data field.
    outputs=wf.steps.JobOutput(name="train_url",
                                obs_config=wf.data.OBSOutputConfig(obs_path=storage.join("directory_path"))),
    # JobStep output
    spec=wf.steps.JobSpec(
        resource=wf.steps.JobResource(
            flavor=wf.Placeholder(name="train_flavor", placeholder_type=wf.PlaceholderType.JSON,
description="Training flavor")
        )
    ), # Training flavors
    depend_steps=[condition_step]
)

# Use JobStep to define a training phase, and use OBS to store the output.
job_step_2 = wf.steps.JobStep(
    name="training_job_2", # Name of a training phase. The name contains a maximum of 64 characters,
including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and must be
unique in a workflow.
    title="Image classification training", # Title, which defaults to the value of name.
    algorithm=wf.AIGalleryAlgorithm(
        subscription_id="subscription_id", # Subscription ID of the subscribed algorithm
        item_version_id="item_version_id", # Algorithm version ID. You can also enter the version number
instead.
        parameters=[]
    ), # Algorithm used for training. An algorithm subscribed to in AI Gallery is used in this example. If the
value of an algorithm hyperparameter does not need to be changed, you do not need to configure the
hyperparameter in parameters. Hyperparameter values will be automatically filled.

    inputs=wf.steps.JobInput(name="data_url", data=obs_data),
    # JobStep input is configured when the workflow is running. You can also use
data=wf.data.OBSPath(obs_path="fake_obs_path") for the data field.
    outputs=wf.steps.JobOutput(name="train_url",
                                obs_config=wf.data.OBSOutputConfig(obs_path=storage.join("directory_path"))),
    # JobStep output
    spec=wf.steps.JobSpec(
        resource=wf.steps.JobResource(
            flavor=wf.Placeholder(name="train_flavor", placeholder_type=wf.PlaceholderType.JSON,
description="Training flavor")
        )
    ), # Training flavors
    depend_steps=[condition_step]
)

# Define model name parameters.
model_name = wf.Placeholder(name="placeholder_name", placeholder_type=wf.PlaceholderType.STR)

model_step = wf.steps.ModelStep(
    name="model_registration", # Name of the model registration phase. The name contains a maximum of
64 characters, including only letters, digits, underscores (_), and hyphens (-). It must start with a letter and
must be unique in a workflow.
    title="Model Registration", # Title
    inputs=wf.steps.ModelInput(name='model_input',
data=wf.data.DataConsumptionSelector(data_list=[job_step_1.outputs["train_url"].as_input(),
job_step_2.outputs["train_url"].as_input()])), # Select the output of job_step_1 or job_step_2 as the input.
    outputs=wf.steps.ModelOutput(name='model_output',
model_config=wf.steps.ModelConfig(model_name=model_name, model_type="TensorFlow")), # ModelStep
outputs
    depend_steps=[job_step_1, job_step_2] # Preceding job phase
)# job_step is an instance object of wf.steps.JobStep and train_url is the value of the name field of
wf.steps.JobOutput.
```

```
workflow = wf.Workflow(name="data-select-demo",
    desc="this is a test workflow",
    steps=[condition_step, job_step_1, job_step_2, model_step],
    storages=storage
)
```

NOTE

The workflow in this example has two parallel branches, but only one branch runs at a time, depending on the configuration of **condition_step**. The input source of **model_step** is either **job_step_1** or **job_step_2**'s output. If **job_step_1** runs and **job_step_2** is skipped, **model_step** uses **job_step_1**'s output as input, and vice versa.

4.3.6 Creating a Workflow

To create a workflow, define each phase by referring to [Creating Workflow Phases](#). Follow these steps:

1. Sort out scenarios, understand preset steps' functions, and determine the DAG structure.
2. Debug single-phase functions like training or inference on ModelArts.
3. Choose the code template that matches the phase function and fill in the details.
4. Arrange phases according to the DAG structure to create a workflow.

Importing the Workflow Data Package

When creating a workflow, required objects are imported through workflow packages. The details are as follows:

```
from modelarts import workflow as wf
```

Import the data package:

```
wf.data.DatasetTypeEnum
wf.data.Dataset
wf.data.DatasetVersionConfig
wf.data.DatasetPlaceholder
wf.data.ServiceInputPlaceholder
wf.data.ServiceData
wf.data.ServiceUpdatePlaceholder
wf.data.DataTypeEnum
wf.data.ModelData
wf.data.GalleryModel
wf.data.OBSPath
wf.data.OBSOutputConfig
wf.data.OBSPlaceholder
wf.data.SWRImage
wf.data.SWRImagePlaceholder
wf.data.Storage
wf.data.InputStorage
wf.data.OutputStorage
wf.data.LabelTask
wf.data.LabelTaskPlaceholder
wf.data.LabelTaskConfig
wf.data.LabelTaskTypeEnum
wf.data.MetricsConfig
wf.data.TripartiteServiceConfig
wf.data.DataConsumptionSelector
```

Import the policy package:

```
wf.policy.Policy  
wf.policy.Scene
```

Import the steps package:

```
wf.steps.MetricInfo  
wf.steps.Condition  
wf.steps.ConditionTypeEnum  
wf.steps.ConditionStep  
wf.steps.LabelingStep  
wf.steps.LabelingInput  
wf.steps.LabelingOutput  
wf.steps.LabelTaskProperties  
wf.steps.ImportDataInfo  
wf.steps.DataOriginTypeEnum  
wf.steps.DatasetImportStep  
wf.steps.DatasetImportInput  
wf.steps.DatasetImportOutput  
wf.steps.AnnotationFormatConfig  
wf.steps.AnnotationFormatParameters  
wf.steps.AnnotationFormatEnum  
wf.steps.Label  
wf.steps.ImportTypeEnum  
wf.steps.LabelFormat  
wf.steps.LabelTypeEnum  
wf.steps.ReleaseDatasetStep  
wf.steps.ReleaseDatasetInput  
wf.steps.ReleaseDatasetOutput  
wf.steps.CreateDatasetStep  
wf.steps.CreateDatasetInput  
wf.steps.CreateDatasetOutput  
wf.steps.DatasetProperties  
wf.steps.SchemaField  
wf.steps.ImportConfig  
wf.steps.JobStep  
wf.steps.JobMetadata  
wf.steps.JobSpec  
wf.steps.JobResource  
wf.steps.JobTypeEnum  
wf.steps.JobEngine  
wf.steps.JobInput  
wf.steps.JobOutput  
wf.steps.LogExportPath  
wf.steps.MrsJobStep  
wf.steps.MrsJobInput  
wf.steps.MrsJobOutput  
wf.steps.MrsJobAlgorithm  
wf.steps.ModelStep  
wf.steps.ModelInput  
wf.steps.ModelOutput  
wf.steps.ModelConfig  
wf.steps.Template  
wf.steps.TemplateInputs  
wf.steps.ServiceStep  
wf.steps.ServiceInput  
wf.steps.ServiceOutput  
wf.steps.ServiceConfig  
wf.steps.StepPolicy
```

Import the workflow package:

```
wf.workflow  
wf.Subgraph  
wf.Placeholder  
wf.PlaceholderType  
wf.AlgorithmParameters  
wf.BaseAlgorithm  
wf.Algorithm  
wf.AIGalleryAlgorithm  
wf.resource
```



```
wf.SystemEnv  
wf.add_whitelist_users  
wf.delete_whitelist_users
```

4.3.7 Publishing a Workflow

4.3.7.1 Publishing a Workflow to ModelArts

You can publish a workflow to ModelArts in two ways: **Publishing to the Running State** and **Publishing and Executing the Workflow**. Publishing to the running state requires configuring input and output parameters on the workflow page. Publishing and executing the workflow allows you to modify the code and run it directly through the SDK, eliminating the need to configure and run it on the console.

Publishing to the Running State

After creating a workflow, you can use the **release()** method to publish the workflow to the running state for configuration and execution (on the workflow page of the management console).

Run the following command:

```
workflow.release()
```

After the preceding command is executed, if the log indicates that the workflow is published, you can go to the ModelArts workflow page to view the workflow. On the workflow details page, click **Configure** to configure parameters.

The **release_and_run()** method is based on the **release()** method and allows you to publish and run workflows in the development state, without the need to configure and execute workflows on the console.



CAUTION

Note the following when using this method:

- For all configuration objects related to placeholders in the workflow, you need to either set default values or use fixed data objects directly.
- The method executes differently depending on the workflow object's name. It creates and runs a new workflow if the name does not exist. It updates and runs the existing workflow if the name already exists, using the new workflow structure for the new execution.

```
workflow.release_and_run()
```

Publishing and Executing the Workflow

With this method, you can publish and run workflows on the SDK without using the console. You need to modify the workflow code as follows:

```
from modelarts import workflow as wf  
  
# Define a unified output storage path.  
output_storage = wf.data.OutputStorage(name="output_storage", description="Unified configuration of  
output storage", default="**")
```

```
# Dataset object
dataset = wf.data.DatasetPlaceholder(name="input_data", default=wf.data.Dataset(dataset_name="***",
version_name="***"))

# Create a training job.
job_step = wf.steps.JobStep(
    name="training_job",
    title="Image Classification Training",
    algorithm=wf.AIGalleryAlgorithm(
        subscription_id="***", # Subscription ID of the image classification algorithm. Obtain the subscription
ID on the algorithm management page. This parameter is optional.
        item_version_id="10.0.0", # Version number of the subscribed algorithm. This parameter is optional.
        parameters=[
            wf.AlgorithmParameters(name="task_type", value="image_classification_v2"),
            wf.AlgorithmParameters(name="model_name", value="resnet_v1_50"),
            wf.AlgorithmParameters(name="do_train", value="True"),
            wf.AlgorithmParameters(name="do_eval_along_train", value="True"),
            wf.AlgorithmParameters(name="variable_update", value="horovod"),
            wf.AlgorithmParameters(name="learning_rate_strategy",
value=wf.Placeholder(name="learning_rate_strategy", placeholder_type=wf.PlaceholderType.STR,
default="0.002", description="Learning rate for training. 10:0.001,20:0.0001 indicates that the learning rate
of the first 10 epochs is 0.001 and that of the next 10 epochs is 0.0001. If the epoch is not specified, the
learning rate will be adjusted based on the validation precision. The training will be stopped if the precision
is not significantly improved anymore.")),
            wf.AlgorithmParameters(name="batch_size", value=wf.Placeholder(name="batch_size",
placeholder_type=wf.PlaceholderType.INT, default=64, description="Number of images trained in each step
(on a single card)")),
            wf.AlgorithmParameters(name="eval_batch_size", value=wf.Placeholder(name="eval_batch_size",
placeholder_type=wf.PlaceholderType.INT, default=64, description="Number of images validated in each
step (on a single card)")),
            wf.AlgorithmParameters(name="evaluate_every_n_epochs",
value=wf.Placeholder(name="evaluate_every_n_epochs", placeholder_type=wf.PlaceholderType.FLOAT,
default=1.0, description="Validation is performed every n epochs.")),
            wf.AlgorithmParameters(name="save_model_secs", value=wf.Placeholder(name="save_model_secs",
placeholder_type=wf.PlaceholderType.INT, default=60, description="Model saving frequency (unit: s)")),
            wf.AlgorithmParameters(name="save_summary_steps",
value=wf.Placeholder(name="save_summary_steps", placeholder_type=wf.PlaceholderType.INT, default=10,
description="Summary saving frequency (unit: step)")),
            wf.AlgorithmParameters(name="log_every_n_steps",
value=wf.Placeholder(name="log_every_n_steps", placeholder_type=wf.PlaceholderType.INT, default=10,
description="Log printing frequency (unit: step)")),
            wf.AlgorithmParameters(name="do_data_cleaning",
value=wf.Placeholder(name="do_data_cleaning", placeholder_type=wf.PlaceholderType.STR, default="True",
description="Whether to clean data. If the data format is abnormal, the training fails. You are advised to
enable this function to ensure training stability. If the data volume is too large, data cleaning may take a
long time. You can clean data offline. (Formats including BMP, JPEG, PNG, and RGB three-channel are
supported.) You are advised to use JPEG.")),
            wf.AlgorithmParameters(name="use_fp16", value=wf.Placeholder(name="use_fp16",
placeholder_type=wf.PlaceholderType.STR, default="True", description="Whether to use mixed precision.
Mixed precision accelerates training but causes precision loss. Enable this parameter unless precision is
strictly required.")),
            wf.AlgorithmParameters(name="xla_compile", value=wf.Placeholder(name="xla_compile",
placeholder_type=wf.PlaceholderType.STR, default="True", description="Whether to use XLA for accelerated
training. This function is enabled by default.")),
            wf.AlgorithmParameters(name="data_format", value=wf.Placeholder(name="data_format",
placeholder_type=wf.PlaceholderType.ENUM, default="NCHW", enum_list=["NCHW", "NHWC"],
description="Input data format. NHWC indicates channel last, and NCHW indicates channel first. This
parameter defaults to NCHW (faster).")),
            wf.AlgorithmParameters(name="best_model", value=wf.Placeholder(name="best_model",
placeholder_type=wf.PlaceholderType.STR, default="True", description="Whether to save and use the model
with the highest precision instead of the latest model during training. The default value is True, indicating
that the optimal model is saved. Within a certain error range, the latest high precision model is saved as
the optimal model.")),
            wf.AlgorithmParameters(name="jpeg_preprocess", value=wf.Placeholder(name="jpeg_preprocess",
placeholder_type=wf.PlaceholderType.STR, default="True", description="Whether to use the JPEG
preprocessing acceleration operator (only JPEG data is supported) to accelerate data reading and improve
performance. This function is enabled by default. If the data format is not JPEG, enable data cleaning to use
the function."))
        ]
    )
```

```
]
),
inputs=[wf.steps.JobInput(name="data_url", data=dataset)],
outputs=[wf.steps.JobOutput(name="train_url",
obs_config=wf.data.OBSOutputConfig(obs_path=output_storage.join("/train_output/")))],
spec=wf.steps.JobSpec(
    resource=wf.steps.JobResource(
        flavor=wf.Placeholder(
            name="training_flavor",
            placeholder_type=wf.PlaceholderType.JSON,
            description="Training flavor",
            default={"flavor_id": "***"}
        )
    )
)
)
)

# Create a workflow.
workflow = wf.Workflow(
    name="image-classification-ResNeSt",
    desc="this is an image classification workflow",
    steps=[job_step],
    storages=[output_storage]
)
```

1. Fill in the actual values for all ** in the code above. The configuration mainly involves these three items:
 - Unified storage: default value of **output_storage**. Enter an existing OBS path in the format of */OBS bucket name/Folder path/*.
 - Dataset object: Enter the dataset name and version number.
 - Training flavor: Configure GPU resources since the algorithm in this example can run only on GPUs. You can use free flavor **modelarts.p3.large.public.free**.
2. After the configuration, run this code:

```
workflow.release_and_run()
```
3. After the execution, go to the ModelArts console. In the navigation pane, choose **Workflow** to view the workflow status.

4.3.7.2 Publishing a Workflow to AI Gallery

You can publish workflows to AI Gallery and share them with other users. To do so, run this code:

```
workflow.release_to_gallery()
```

Once the release is done, you can visit AI Gallery to see the asset details. The asset permission is set to private by default, but you can change it if you want.

1. Go to AI Gallery.
2. Choose **My Gallery > My Assets > Workflow**.
3. In the **My Publishes** tab, view the workflow published to AI Gallery.
4. Click the workflow name to view the workflow details.

The **release_to_gallery()** method contains the following input parameters.

Parameter	Description	Mandatory	Type
content_id	Workflow asset ID	No	str

Parameter	Description	Mandatory	Type
version	Version number of the workflow asset. The format is <i>x.x.x</i> .	No	str
desc	Description of the workflow asset version	No	str
title	Workflow asset name. If this parameter is not specified, the workflow name is used by default.	No	str
visibility	Visibility of the workflow asset. The value can be public , group (whitelist), and private (visible only to you). The default value is private .	No	str
group_users	Whitelist. This parameter is mandatory only when visibility is set to group . You can only enter domain_id .	No	list[str]

You can use the method in two ways based on the input parameters:

- `workflow.release_to_gallery(title="Asset name")` publishes a new workflow asset with version 1.0.0. If the workflow includes an algorithm that is not from AI Gallery, it also publishes the algorithm to AI Gallery with version 1.0.0.
- `workflow.release_to_gallery(content_id="**", title="Asset name")` publishes a new version based on the specified workflow asset. The version number increases automatically. If the workflow includes an AI Gallery algorithm, it updates the algorithm asset with a higher version number.

Workflow asset whitelist setting:

You can specify the **visibility** and **group_users** fields of the **release_to_gallery** method when you publish an asset for the first time. To change the whitelist for accessing a certain asset, use this command:

```
from modelarts import workflow as wf

# Add specified whitelist users.
wf.add_whitelist_users(content_id="**", version_num="*.*", user_groups=["**", "**"])

# Delete specified whitelist users.
wf.delete_whitelist_users(content_id="**", version_num="*.*", user_groups=["**", "**"])
```

 NOTE

When you modify the whitelist for a workflow asset, the system automatically obtains the information of the algorithm asset that the workflow version depends on and sets the whitelist for the algorithm asset as well.

4.3.8 Advanced Workflow Capabilities

4.3.8.1 Using Big Data Capabilities (MRS) in a Workflow

Function

This phase calls MRS for big data cluster computing. It is used for batch data processing and model training.

Application Scenarios

You can use MRS Spark for big data computing in this phase.

Examples

On the Huawei Cloud MRS console, check available MRS clusters of your account. If no MRS cluster is available, create one with Spark selected.

You can use MrsStep to create a job phase. The following is an example of defining a MrsStep:

- **Specifying a boot script and cluster**

```
from modelarts import workflow as wf
# Define a MrsJobStep using MrsStep.

algorithm = wf.steps.MrsJobAlgorithm(
    boot_file="obs://spark-sql/wordcount.py", # OBS path to the boot script
    parameters=[wf.AlgorithmParameters(name="run_args", value="--master,yarn-cluster")]
)
inputs = wf.steps.MrsJobInput(name="mrs_input", data=wf.data.OBSPath(obs_path="/spark-sql/
mrs_input/")) # OBS path to the input data
outputs = wf.steps.MrsJobOutput(name="mrs_output",
obs_config=wf.data.OBSOutputConfig(obs_path="/spark-sql/mrs_output")) # OBS path to the output
data
step = wf.steps.MrsJobStep(
    name="mrs_test", # Step name, which can be customized
    mrs_algorithm=algorithm,
    inputs=inputs,
    outputs=outputs,
    cluster_id="cluster_id_xxx" # MRS cluster ID
)
```

- **Configuring a cluster and boot script**

```
from modelarts import workflow as wf
# Define a phase using MrsJobStep.
run_arg_description = "Program execution parameter, which is used as the program running
environment parameter. The default value is (--master,yarn-cluster)".
app_arg_description = "Program execution parameter, which is used as the input parameter of the
boot script, for example, (--param_a=3,--param_b=4). This parameter is optional and left blank by
default."
mrs_outputs_description = "Data output path, which can be obtained from train_url in the parameter
list."
cluster_id_description = "cluster id of MapReduce Service"
```

```
algorithm = wf.steps.MrsJobAlgorithm(  
    boot_file=wf.Placeholder(name="boot_file",  
        description="Program boot script",  
        placeholder_type=wf.PlaceholderType.STR,  
        placeholder_format="obs"),  
    parameters=[wf.AlgorithmParameters(name="run_args",  
        value=wf.Placeholder(name="run_args",  
            description=run_arg_description,  
            default="--master,yarn-cluster",  
            placeholder_type=wf.PlaceholderType.STR),  
        ),  
        wf.AlgorithmParameters(name="app_args",  
            value=wf.Placeholder(name="app_args",  
                description=app_arg_description,  
                default="",  
                placeholder_type=wf.PlaceholderType.STR)  
        )  
    ]  
)  
  
inputs = wf.steps.MrsJobInput(name="data_url",  
    data=wf.data.OBSPlaceholder(name="data_url",object_type="directory"))  
  
outputs = wf.steps.MrsJobOutput(name="train_url",  
    obs_config=wf.data.OBSOutputConfig(obs_path=wf.Placeholder(name="train_url",  
        placeholder_type=wf.PlaceholderType.STR,  
        placeholder_format="obs",description=mrs_outputs_description)))  
  
mrs_job_step = wf.steps.MrsJobStep(  
    name="mrs_job_test",  
    mrs_algorithm=algorithm,  
    inputs=inputs,  
    outputs=outputs,  
    cluster_id=wf.Placeholder(name="cluster_id", placeholder_type=wf.PlaceholderType.STR,  
        description=cluster_id_description, placeholder_format="cluster")  
)
```

- **Using an MRS phase on the console**

After a workflow is published, configure phase parameters such as the data input, data output, boot script, and cluster ID on the workflow configuration page.

4.3.8.2 Specifying Certain Phases to Run in a Workflow

Workflows support predefined scenarios to enable partial execution. You can split the DAG into different branches based on the scenarios during workflow development. Then, you can run each branch independently as a separate workflow. The sample code is as follows:

```
workflow = wf.Workflow(  
    name="image_cls",  
    desc="this is a demo workflow",  
    steps=[label_step, release_data_step, training_step, model_step, service_step],  
    policy=wf.policy.Policy(  
        scenes=[  
            wf.policy.Scene(  
                scene_name="Model training",  
                scene_steps=[label_step, release_data_step, training_step]  
            ),  
            wf.policy.Scene(  
                scene_name="Service deployment",  
                scene_steps=[model_step, service_step]  
            ),  
        ]  
    )  
)
```

This example shows a workflow with five phases. The policy defines two preset scenarios: model training and service deployment. When the workflow is published to the running state, partial execution is disabled by default and all phases run. You can specify certain scenarios to run on the global configuration page.

 NOTE

You can define the same phase in different running scenarios using partial execution. However, you must ensure that the data dependency between phases is correct. Partial execution can only be configured and used in the running state and cannot be debugged in the development state.

5 Using Notebook for AI Development and Debugging

5.1 Application Scenarios

ModelArts provides flexible, open development environments. Select a development environment based on site requirements.

- In-cloud notebook, which is out of the box, relieving you from concerning environment installation or configuration. For details, see [Creating a Notebook Instance \(Default Page\)](#).
- ModelArts Notebook supports the following methods to develop AI models based on engines such as PyTorch, TensorFlow, and MindSpore:
 - Use JupyterLab to open a notebook instance. For details, see [Using a Notebook Instance for AI Development Through JupyterLab](#).
 - Local IDE for model development. After enabling remote SSH, you can remotely access the ModelArts notebook development environment to debug and run code from a local IDE. The local IDE allows you to use the in-cloud notebook development environment while with local coding habits unchanged.

A local IDE supports Visual Studio (VS) Code, PyCharm, and SSH. Additionally, PyCharm Toolkit and VS Code Toolkit are provided for convenient remote access. For details, see [Using Notebook Instances Remotely Through PyCharm](#), [Using Notebook Instances Remotely Through VS Code](#), and [Using a Notebook Instance Remotely with SSH](#).

- Easy and fast file uploading is a common requirement in AI development. ModelArts allows you to upload files in multiple ways. You can view the upload progress and speed during the uploading.
 - [Upload local files](#).
 - [Upload GitHub open-source repository files](#).
 - [Upload OBS files](#).
 - [Upload remote files such as open-source datasets](#).
- When using a notebook instance, you can quickly obtain target instances and switch images in the same instance. You can flexibly adjust the specifications

by adjusting the AI engine and switching node runtime specifications. For users who just start to use ModelArts notebook, choose a rather small storage at first. After the notebook instance is created, scale out EVS as needed and use dynamic mounting to simulate OBS objects as a local file system. You can also view events to locate faults when a notebook instance is faulty. For details, see [Managing Notebook Instances](#).

- ModelArts CLI (ma-cli) is integrated in the notebook instance to interconnect with ModelArts and run management commands on ModelArts resources. ma-cli allows you to interact with cloud services in ModelArts notebook instances and offline VMs. You can run ma-cli commands to implement automatic command completion, authentication, image building, ModelArts training job submission, DLI Spark job submission, and OBS data replication. For details, see [ModelArts CLI Command Reference](#).
- ModelArts Notebook has a built-in MoXing Framework module. ModelArts mox.file provides a set of APIs for accessing OBS more conveniently, allowing users to operate OBS files by simulating a series of APIs for operating the local file system. For details, see [Using MoXing Commands in a Notebook Instance](#).

5.2 Creating a Notebook Instance (Default Page)

Before developing a model, create a notebook instance and access it for coding.

Context

- Notebook is billed as follows:
 - A running notebook instance will be billed based on used resources. The fees vary depending on your selected resources. For details, see [Pricing Details](#). When a notebook instance is not used, stop it.
 - If you select EVS for storage when creating a notebook instance, the EVS disk will be continuously billed if the instance is not deleted. Stop and delete the notebook instance if it is not required. For more information, see [Development Environment](#).
- When a notebook instance is created, auto stop is enabled by default. The notebook instance will automatically stop at the specified time.
- Only running notebook instances can be accessed or stopped.
- A maximum of 10 notebook instances can be created under one account.

Procedure

1. Log in to the [ModelArts console](#). In the navigation pane on the left, choose **Permission Management** and check whether the access authorization has been configured. If not, configure access authorization. For details, see [Configuring Agency Authorization for ModelArts with One Click](#).

Figure 5-1 Viewing agency configurations

Authorized To	Authorized User	Authorization Type	Authorization Content	Creation Time	Operation
all-users	All users	Agency	modelarts_agency_allbb	Mar 06, 2024 10:27:12 GMT+08:00	View Permissions Delete

2. Log in to the [ModelArts console](#). In the navigation pane on the left, choose **Development Workspace > Notebook**.

3. Click **Create Notebook** in the upper right corner. On the displayed page, configure the parameters.
- a. Configure the basic information of the notebook instance, including its name, description, and auto stop status. For details, see [Table 5-1](#).

Figure 5-2 Basic information of a notebook instance

★ Name

notebook-38f6

Description

0/512

★ Auto Stop

☒

1

Enable this option to specify a time for the notebook instance to automatically stop. You will not be billed after it has stopped.

Table 5-1 Basic parameters

Parameter	Description
Name	Name of the notebook instance, which is automatically generated by the system. You can rename it based on service requirements. A name consists of a maximum of 128 characters and cannot be empty. It can contain only digits, letters, underscores (_), and hyphens (-).
Description	Brief description of the notebook instance
Auto Stop	Automatically stops the notebook instance at a specified time. This function is enabled by default. The default value is 1 hour, indicating that the notebook instance automatically stops after running for 1 hour and its resource billing will stop then. The options are 1 hour, 2 hours, 4 hours, 6 hours, and Custom. You can select Custom to specify any integer from 1 to 72 hours. • Stop as scheduled: If this option is enabled, the notebook instance automatically stops when the running duration exceeds the specified duration. NOTE To protect in-progress jobs, a notebook instance does not automatically stop immediately at the auto stop time. Instead, there is a 2 to 5 minutes delay for you to renew the auto stop time.

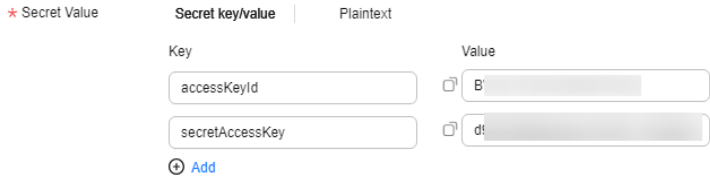
- b. Configure notebook parameters, such as the image and instance flavor. For details, see [Table 5-2](#).

Table 5-2 Notebook instance parameters

Parameter	Description
Image	Public and private images are supported. • Public images are the AI engines built in ModelArts. • You can use a custom image created by yourself. Use either of the following ways to create a custom image: – Save the instance created using the public image and use it as a custom image. For details, see Saving a Notebook Instance. – Create a custom image using a preset or third-party image. Custom images must meet the image specifications. After the image is created, register it on the image management page of ModelArts before using it in a notebook instance. For details, see Creating a Custom Image. An image corresponds to an AI engine. When you select an image during instance creation, the AI engine is specified accordingly. Select an image as required. Enter a keyword of the image name in the search box on the right to quickly search for the image. You can change an image on a stopped notebook instance.
JupyterLab	JupyterLab supports the following two versions. The default version is 3.2.3. User experience and version stability vary among the two versions. • 3.2.3: A stable version that has been widely verified and is suitable for long-term use. Select this version if you need a stable working environment. • 4.3.1: The latest version that includes multiple new functions and improvements, but may be unstable. Select this version to experience the latest functions. For details, see JupyterLab 4.3.1. NOTE JupyterLab 3.2.3 will be gradually brought offline. Select a version as required.

Parameter	Description
Resource Type	Public and dedicated resource pools are available. For dedicated resource pools, you can select heterogeneous resource pools of CPUs, NPUs, and GPUs. For example, if the node flavor supports CPUs and GPUs, you can select CPUs or GPUs as the instance specifications. Public resource pools are billed based on the running duration of your notebook instances. Select a created dedicated resource pool based on site requirements. If no dedicated resources are available, purchase one. NOTE If the dedicated resource pool you purchased is a single-node Tnt004 pool whose specification is GPU: 1*tnt004 CPU: 8 vCPUs and 32 GiB (modelarts.vm.gpu._tnt004u8), when you use the cluster to create a notebook instance, the Tnt004 card is idle but is displayed as sold out or the creation fails due to insufficient resources, contact technical support.
Type	Processor type, which can be CPU or GPU. The chips vary depending on the selected image. GPUs deliver better performance than CPUs but at a higher cost. Select a chip type as needed.
Instance Specifications	The available resource specifications vary among chip types. Select the specifications based on your needs. • CPU 2vCPUs 8GB: General-purpose Intel CPU flavor, ideal for rapid data exploration and experiments 8vCPUs 32GB: General computing-plus Intel CPU flavor, ideal for compute-intensive applications • GPU GPU: 1*Vnt1(32GB) CPU: 8vCPUs 64GB: Single GPU with 32 GB of memory, ideal for algorithm training and debugging in deep learning scenarios GPU: 1*Tnt004(16GB) CPU: 8vCPUs* 32GB: Single GPU with 16 GB of memory, ideal for inference computing such as computer vision, video processing, and NLP tasks GPU: 1*Pnt1(16GB) CPU: 8vCPUs 64GB: Single GPU with 16 GB of memory, ideal for algorithm training and debugging in deep learning scenarios

Parameter	Description
Storage	The value can be EVS, SFS, OBS, or PFS. Configure this parameter based on your needs. NOTE OBS and PFS are whitelist functions. If you have trial requirements, submit a service ticket to apply for permissions. EVS Set a disk size based on service requirements. The default value is 5 GB. The maximum disk size is displayed on the GUI. The EVS disk space is charged by GB from the time the notebook instance is created to the time the notebook instance is deleted. SFS Select this type only for a dedicated resource pool. SFS takes effect only after a dedicated resource pool can communicate with your VPC. For details, see Creating a Network. NOTE For details about how to set permissions to access SFS Turbo folders, see Permissions Management. SFS: Select the created SFS Turbo file system (created on the SFS console). Cloud Mount Path: Retain the default value /home/ma-user/work/. Mounted Subdirectory: Select the storage path on SFS Turbo. Mount Method: This parameter is displayed when the folder control permission is granted for the user. The read/write or read-only permission is displayed based on the storage path on SFS Turbo. - The value can be OBS or PFS. Storage Path: Set the OBS path for storing notebook data. If you want to use existing files or data, upload them to the specified OBS path. Storage Path must be set to a specific directory in an OBS bucket rather than the root directory of the OBS bucket. Secret: Select an existing secret or click Create on the right to create one. On the displayed DEW console, create a secret. Enter accessKeyId and secretAccessKey under Key, and enter the AKs/SKs obtained from My Credentials > Access Keys under Value.

Parameter	Description
	Figure 5-3 Configuring the secret values  EVS and SFS are all mounted to the /home/ma-user/work directory. You can add a data storage path during the runtime of a notebook instance by referring to Dynamically Mounting an OBS Parallel File System. The data is retained in /home/ma-user/work, even if the notebook instance is stopped or restarted. When a notebook instance is deleted, the EVS storage is released and the stored data is not retained. SFS can be mounted to a new notebook instance and data can be retained.
Extended Storage	NOTE This parameter is a whitelist function. If you have trial requirements, submit a service ticket to apply for permissions. If you need multiple data storage paths, click Add Extended Storage to add more storage mount directories. You can add an OBS, PFS, or SFS directory. Constraints: - For each type, a maximum of five directories can be mounted. - The directories must be unique and cannot be mounted to a blacklisted directory. Nested mounting is allowed. Blacklisted directories are those with the following prefixes: /data/, /cache/, /dev/, /etc/, /bin/, /lib/, /sbin/, /modelarts/, /train-worker1-log/, /var/, /resource_info/, /usr/, /sys/, /run/, /tmp/, /infer/, and /opt/ After this parameter is configured, the notebook instance details page is displayed. Click Storage > Extended Storage to view or edit the extended storage information. If the number of storage devices does not reach the maximum, you can click Add Extended Storage.

Parameter	Description
Remote SSH	- After you enable this function, you can remotely access the development environment of the notebook instance from your local development environment. - When a notebook instance is stopped, you can update the SSH configuration on the instance details page. NOTE The notebook instances with remote SSH enabled have VS Code plug-ins (such as Python and Jupyter) and the VS Code server package pre-installed, which occupy about 1 GB persistent storage space.
Key Pair	Set a key pair after remote SSH is enabled. Select an existing key pair. Alternatively, click Create on the right of the text box to create one on the DEW console. On the displayed Create Account Key Pair page, configure the parameters. After a notebook instance is created, you can change the key pair on the instance details page. CAUTION Download the created key pair and properly keep it. When you use a local IDE to remotely access the notebook development environment, the key pair is required for authentication.
Whitelist	Set a whitelist after remote SSH is enabled. This parameter is optional. Add the IP addresses for remotely accessing the notebook instance to the whitelist, for example, the IP address of your local PC or the public IP address of the source device. A maximum of five IP addresses can be added and separated by commas (,). If the parameter is left blank, all IP addresses will be allowed for remote SSH access. If your source device and ModelArts are isolated from each other in network, obtain the public IP address of your source device using a mainstream search engine, for example, by entering "IP address lookup", but not by running ipconfig or ifconfig/ip locally. After a notebook instance is created, you can change the whitelist IP addresses on the instance details page.

- c. (Optional) Add tags to the notebook instance. Enter a tag key and value and click **Add**.

Table 5-3 Adding a tag

Parameter	Description
Tags	ModelArts can work with Tag Management Service (TMS). When creating resource-consuming tasks in ModelArts, for example, training jobs, configure tags for these tasks so that ModelArts can use tags to manage resources by group. For details about how to use tags, see How Does ModelArts Use Tags to Manage Resources by Group? After adding a tag, you can view, modify, or delete the tag on the notebook instance details page.

 NOTE

You can select a predefined TMS tag from the tag drop-down list or customize a tag. Predefined tags are available to all service resources that support tags. Customized tags are available only to the service resources of the user who has created the tags.

- Click **Next**.
- After confirming the parameter settings, click **Submit**.
Switch to the notebook instance list. The notebook instance is being created. It will take several minutes before its status changes to **Running**. Then, the notebook instance is created.
- In the notebook instance list, click the instance name. On the instance details page that is displayed, view the instance configuration.

If **Remote SSH** is enabled, you can click the modification icon on the right of the whitelist to modify it. You can click the modification icon on the right of **Authentication** to update the key pair of a stopped notebook instance.

In the **Storage** tab, click **Mount Storage** to mount an OBS parallel file system to the instance for reading data. For details, see [Dynamically Mounting an OBS Parallel File System](#).

If an EVS disk is used, click **Expansion** on the right of **Storage Capacity** to dynamically expand the EVS disk capacity. For details, see [Dynamically Expanding EVS Disk Capacity](#).

Accessing a Notebook Instance

Access a notebook instance in the **Running** state for coding.

- Online access: Use JupyterLab. For details, see [Using a Notebook Instance for AI Development Through JupyterLab](#).

- Remotely accessed from a local IDE through PyCharm. For details, see [Using Notebook Instances Remotely Through PyCharm](#).
- Remotely accessed from a local IDE through VS Code. For details, see [Using Notebook Instances Remotely Through VS Code](#).
- Remotely accessed from a local IDE through SSH. For details, see [Using a Notebook Instance Remotely with SSH](#).

ModelArts notebook instances are started by `ma-user` by default. After you access the instance, the default working directory is `/home/ma-user/work`.

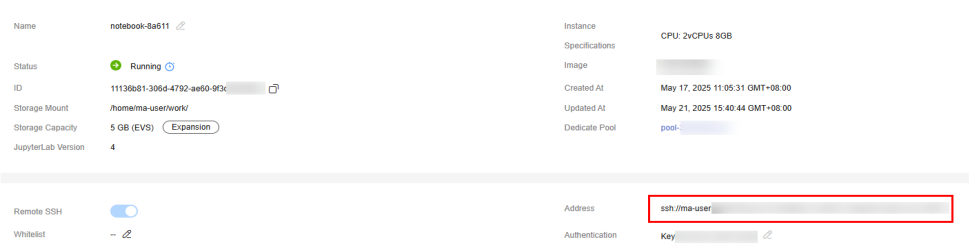
Figure 5-4 Working directory example



Most notebook instances in a dedicated resource pool are started as user `root`. The details are as follows:

- When you log in to the terminal as user `root`, the system automatically runs the `source /home/ma-user/.bashrc` command to synchronize the environment variables of user `ma-user`. To disable this function, set the environment variable `export DISABLE_MA_USER_BASHRC` to `true` in the custom image to prevent the `/home/ma-user/.bashrc` file from being loaded.
- If the instance is started by user `root`, only user `root` can be used for SSH remote connection.

Figure 5-5 Using user `root` for SSH remote connection



JupyterLab 4.3.1

This version has significantly improved user experience, functions, and performance. The following table lists the main updates. For further information, see [JupyterLab Documentation](#).

Table 5-4 Feature updates

Function Module	Description
Workspaces	You can create multiple workspaces to organize and manage different projects and files, improving project management efficiency.
Launcher creation toolbar	Added a toolbar for functional access and convenient operations.
Theme framework optimization	Optimized the theme framework to enhance the theme flexibility and compatibility. You can customize the GUI as required.
Configuration page optimization	Optimized user experience on the configuration page for easier use.
Performance optimization and debugging enhancement	Fully optimized the system performance and enhanced the debugging function to improve the development experience.
Code editor enhancement	Improved the code editor by adding more functions and optimizing performance to improve the code writing and editing experience.
Search function enhancement	Enhanced the search function for quicker and more accurate search.
Performance enhancement	Improved the overall performance, system response speed, and system stability.
Custom CSS style sheets	Custom CSS style sheets can be used to adjust the GUI appearance to meet personalized requirements.
Markdown charts	Charts can be used in Markdown to facilitate document writing.
Virtual scroll bar	Introduced the virtual scroll bar to improve the scrolling experience of large files or a large amount of content.
Workspace UI	Improved the workspace UI to provide better visual effect and operation experience and enhance UI friendliness.
File access records	You can view recently opened and closed files to quickly access common files and improve work efficiency.
Keyboard shortcuts improvement	Keyboard shortcuts are improved for higher operation efficiency and convenience.

Mounting Directories of Notebook Containers

When you use EVS storage when creating a notebook instance, the **/home/ma-user/work** directory is used as the workspace for persistent storage.

The data stored in only the **work** directory is retained after the instance is stopped or restarted. When you use a development environment, store the data for persistence in **/home/ma-user/work**.

For details about directory mounting of a notebook instance, see [Table 5-5](#). The following mounting points are not saved when images are saved.

Table 5-5 Mounting directories

Mount Point	Read Only	Remarks
/home/ma-user/work/	No	Persistent directory of your data
/data	No	Mount directory of your PFS
/cache	No	Used to mount the hard disk of the host NVMe (supported by bare metal specifications)
/train-worker1-log	No	Compatible with training job debugging
/dev/shm	No	Used for PyTorch engine acceleration

Selecting Storage for a Notebook Instance

Storage varies depending on performance, usability, and cost. No storage media can cover all scenarios. Learn about in-cloud storage application scenarios for better usage.

Table 5-6 In-cloud storage application scenarios

Storage	Application Scenario	Advantage	Disadvantage
EVS	Data and algorithm exploration only in the development environment.	Block storage SSDs feature better overall I/O performance than NFS. The storage capacity can be dynamically expanded to up to 4096 GB. As persistent storage, EVS disks are mounted to /home/ma-user/work . The data in this directory is retained after the instance is stopped. The storage capacity can be expanded online based on demand.	This type of storage can only be used in a single development environment.

Storage	Application Scenario	Advantage	Disadvantage
Parallel File System (PFS)	NOTE - PFS is a whitelist function. To use this function, contact Huawei technical support. - Only OBS PFS in the same region can be mounted. PFS buckets mounted as persistent storage for AI development and exploration. - Storage for datasets. Datasets stored in the PFS buckets are directly mounted to notebooks for browsing and data processing and can be directly used during training. Select PFS when creating a notebook instance. After the instance is running, the OBS parallel file system that carries the datasets is dynamically mounted to the notebook instances. For details, see Dynamically Mounting an OBS Parallel File System. - Storage for code. After debugging on a notebook instance, specify the OBS path as the code path for starting training, facilitating	PFS is an optimized high-performance object storage file system with low storage costs and large throughput. It can quickly process high-performance computing (HPC) workloads. PFS mounting is recommended if OBS is used. NOTE Package or split the data to be uploaded by 128 MB or 64 MB. Download and decompress the data in local storage for better I/O and throughput performance.	Due to average performance in frequent read and write of small files, PFS storage is not suitable for large model training or file decompression. NOTE Before mounting PFS storage to a notebook instance, grant ModelArts with full read and write permissions on the PFS bucket. The policy will be retained even after the notebook instance is deleted.

Storage	Application Scenario	Advantage	Disadvantage
	temporary modification. 3. Storage for checking training. Mount storage to the training output path such as the path to training logs. In this way, view and check training on the notebook instance in real time. This is especially suitable for analyzing the output of jobs trained using TensorBoard.		
OBS	NOTE • OBS is a whitelist function. To use this function, contact Huawei technical support. • Only OBS objects in the same region can be mounted. When uploading or downloading a large amount of data in the development environment, you can use OBS buckets to transfer data.	Low storage cost and high throughput, but average performance in reading and writing small files. It is a good practice to package or split the file by 128 MB or 64 MB. In this way, you can download the packages, decompress them, and use them locally.	The object storage semantics is different from the Posix semantics and needs to be further understood.

Storage	Application Scenario	Advantage	Disadvantage
Scalable File Service (SFS)	Available only in dedicated resource pools. Use SFS storage in informal production scenarios such as exploration and experiments. One SFS device can be mounted to both a development environment and a training environment. In this way, you do not need to download data each time your training job starts. This type of storage is not suitable for heavy I/O training on more than 32 cards.	SFS is implemented as NFS and can be shared between multiple development environments and between development and training environments. This type of storage is preferred for non-heavy-duty distributed training jobs, especially for the ones not requiring to download data additionally when the training jobs start.	The performance of the SFS storage is not as good as that of the EVS storage.
OceanStor Pacific storage (SFS capacity-oriented 2.0)	Currently, it can be used only in Tiangong resource pools. It is suitable for the training jobs that use the file systems provided by SFS capacity-oriented 2.0 for AI model training and exploration. In addition, OBS APIs are provided to import training data from outside the cloud.	It provides a high-performance file client to meet the high storage bandwidth requirements of heavy-load training jobs. It also supports access to OBS. After the training data is imported to the storage via OBS APIs, it can be used to train the model directly without any conversion.	It provides the object storage semantics, which is different from the Posix semantics and needs to be further understood.

Storage	Application Scenario	Advantage	Disadvantage
Local storage	First choice for heavy-duty training jobs.	High-performance SSDs for the used VM or BMS, featuring high file I/O throughput. For heavy-duty training jobs, store data in the target directory and then start training. By default, the storage is mounted to the /cache directory. For details about the available space of the /cache directory, see What Are Sizes of the /cache Directories for Different Notebook Specifications in DevEnviron?	The storage lifecycle is associated with the container lifecycle. Data needs to be downloaded each time the training job starts.

FAQ

- How do I use EVS in a development environment?
When creating a notebook instance, select a small-capacity EVS disk. You can scale out the disk as needed. For details, see [Dynamically Expanding EVS Disk Capacity](#).
- How do I use an OBS parallel file system in a development environment?
When training data in a notebook instance, you can use the datasets mounted to a notebook container, and use an OBS parallel file system. For details, see [Dynamically Mounting an OBS Parallel File System](#).
- How do I switch back to JupyterLab 3.2.3 if an error occurs during the startup of JupyterLab 4.3.1?
Locate the target instance in the list and click **Start** in the **Operation** column. In the displayed dialog box, select JupyterLab 3.2.3 and click **OK**.
- Can I use both JupyterLab 3.2.3 and 4.3.1 in a project?
Not recommended. Each JupyterLab instance runs independently. Therefore, you need to create an instance for each version. To try different versions, you can start them in different containers or environments. Pay attention to the following:
 - The configuration file and data path may vary according to the version. Ensure the independence of data and configuration.
 - Running multiple versions at the same time may cause port conflicts or other resource competition problems.
- Can I use GDB in a notebook instance?
No. GDB needs Docker with privileged containers. For security purposes, the development environment does not allow privileged containers. Therefore, GDB cannot be used in notebook instances.

5.3 Creating a Notebook Instance (New Page)

NOTE

ModelArts has enhanced the creation page to improve the efficiency of creating notebook instances. The updated page streamlines operations and enhances the GUI display.

Before developing a model, create a notebook instance and access it for coding.

You can create a notebook instance in either of the following ways:

- Create a notebook instance on the [ModelArts console](#). This section provides the operation guide of the new page. For details about the operation guide of the old (default) page, see [Creating a Notebook Instance \(Default Page\)](#).
- Create a notebook instance via ModelArts APIs. For details, see [Creating a Notebook Instance](#).

Constraints

- Auto stop is enabled by default during notebook instance creation, that is, the notebook instance will automatically stop at the specified time to avoid resource waste.
- Only running notebook instances can be accessed or stopped.
- A maximum of 10 notebook instances can be created under an account.

Billing

Notebook is billed as follows:

- A running notebook instance will be billed based on the used resources. The fees vary depending on your selected resources. For details, see [Pricing Details](#). When a notebook instance is not used, stop it.
- If you select EVS for storage when creating a notebook instance, the EVS disk will be continuously billed if the instance is not deleted. Stop and delete the notebook instance if it is not required. For more information, see [Development Environment](#).

Procedure

1. Log in to the [ModelArts console](#). In the navigation pane on the left, choose **Permission Management** and check whether the access authorization has been configured. If not, configure access authorization. For details, see [Configuring Agency Authorization for ModelArts with One Click](#).

Figure 5-6 Viewing agency configurations

Authorized To 	Authorized User 	Authorization Type 	Authorization Content 	Creation Time 	Operation
all-users	All users	Agency	modelarts_agency_allto	Mar 06, 2024 16:27:12 GMT+08:00	View Permissions Delete

2. Log in to the [ModelArts console](#). In the navigation pane on the left, choose **Development Workspace > Notebook**.
3. Click **Create Notebook** in the upper right corner. On the displayed page, configure the parameters by referring to the following table.

Table 5-7 Parameters for creating a notebook instance

Parameter		Description
Basic Information	Name	Name of the notebook instance, which is automatically generated by the system. You can rename it as required. The name can contain at most 128 characters and cannot be left empty. Only digits, letters, underscores (_), and hyphens (-) are allowed.
	Add Description	Click Add Description to customize the notebook instance description, which cannot exceed 512 characters.
	Tags	To add the same tag to different cloud resources, use Tag Management Service (TMS) to create predefined tags. For details, see Creating Predefined Tags. Click Add and enter a tag key and value. You can add at most 20 tags. After adding a tag, you can view, modify, or delete the tag on the notebook instance details page. NOTE You can select a predefined TMS tag from the tag drop-down list or customize a tag. Predefined tags are available to all service resources that support tags. Customized tags are available only to the service resources of the user who has created the tags.
Auto Stop		This function is enabled by default. The notebook instance tracks its runtime once started. It stops automatically if the runtime surpasses the set time limit. Stop Type: Select Stop as scheduled. In this case, the notebook instance automatically stops when the runtime exceeds the specified time. You can select 1 hour, 2 hours, 4 hours, 6 hours, or Custom. If Custom is selected, enter any integer within the range of 1 to 72. CAUTION To protect in-progress jobs, a notebook instance does not automatically stop immediately at the auto stop time. Instead, there is a 2 to 5 minutes delay for you to renew the auto stop time.

Parameter		Description
Environment	Image	Click Select Image. On the Image panel, select a preset image or a custom image, and click OK. • Preset images are the AI engines built in ModelArts. • You can use a custom image created by yourself. Use either of the following ways to create a custom image: – Save the instance created using the preset image and use it as a custom image. For details, see Saving a Notebook Instance. – Create a custom image using a public or third-party image. Custom images must meet the image specifications. After the image is created, register it on the image management page of ModelArts before using it in a notebook instance. For details, see Creating a Custom Image. An image corresponds to an AI engine. When you select an image during instance creation, the AI engine is specified accordingly. Select an image as required. Enter a keyword of the image name in the search box on the right to quickly search for the image. You can change the image on a stopped notebook instance.
	JupyterLab Version	JupyterLab supports the following two versions. The default version is 3.2.3. User experience and version stability vary among the two versions. • 3.2.3: A stable version that has been widely verified and is suitable for long-term use. Select this version if you need a stable working environment. • 4.3.1: The latest version that includes multiple new functions and improvements, but may be unstable. Select this version to experience the latest functions. For details, see JupyterLab 4.3.1. NOTE JupyterLab 3.2.3 will be gradually brought offline. Select a version based on your requirements.

Parameter		Description
Resource	Resource Pool Type	Public and dedicated resource pools are available. For dedicated resource pools, you can select heterogeneous resource pools of CPUs, NPUs, and GPUs. For example, if the node flavor supports CPUs and GPUs, you can select CPUs or GPUs as the instance specifications. Public resource pools are billed based on the runtime of your notebook instances. Select a created dedicated resource pool based on site requirements. If no dedicated resources are available, purchase one. NOTE If the dedicated resource pool you purchased is a single-node Tnt004 pool whose specification is GPU: 1*tnt004 CPU: 8 vCPUs and 32 GiB (modelarts.vm.gpu._tnt004u8), when you use the cluster to create a notebook instance, the Tnt004 card is idle but is displayed as sold out or the creation fails due to insufficient resources, contact technical support.
	Instance Specifications	Based on the chip you select (CPU or GPU), the available resource specifications vary. Select the specifications based on your needs. - The following shows examples of CPU specifications: 2vCPUs 8GB: General-purpose Intel CPU flavor, ideal for rapid data exploration and experiments. 8vCPUs 32GB: General computing-plus Intel CPU flavor, ideal for compute-intensive applications. - The following shows examples of GPU specifications: GPU: 1*Vnt1(32GB) CPU: 8vCPUs 64GB: Single GPU with 32 GB of memory, ideal for algorithm training and debugging in deep learning scenarios. GPU: 1*Tnt004(16GB) CPU: 8vCPUs* 32GB: Single GPU with 16 GB of memory, ideal for inference computing such as computer vision, video processing, and NLP tasks. GPU: 1*Pnt1(16GB) CPU: 8vCPUs 64GB: Single GPU with 16 GB of memory, ideal for algorithm training and debugging in deep learning scenarios.

Parameter		Description
Storage Settings	Storage Type	Select Elastic Volume Service or Object Storage Service as required. For details about the storage types, see Selecting Storage for a Notebook Instance. NOTE OBS and PFS are whitelist functions. Submit a service ticket for a trial if required. • Elastic Volume Service Set a disk size based on service requirements. The default value is 5 GB. The maximum disk size is displayed on the GUI. The EVS disk space is charged by GB from the time the notebook instance is created to the time the notebook instance is deleted. • Scalable File Service Select this only for a dedicated resource pool. SFS takes effect only after a dedicated resource pool can communicate with your VPC. For details, see Creating a Network. NOTE For details about how to set permissions to access SFS Turbo folders, see Permissions Management. – Scalable File Service: Select a created SFS Turbo file system. To create an SFS Turbo file system, log in to the SFS console. – Cloud Mount Path: Retain the default value <code>/home/ma-user/work/</code>. – Mounted Subdirectory: Select the storage path on SFS Turbo. – Mount Method: This parameter is displayed when the folder control permission is granted for the user. The read/write or read-only permission is displayed based on the storage path on SFS Turbo. • Object Storage Service or Parallel File System Storage Path: Set the OBS path for storing notebook data. If you want to use existing files or data, upload them to the specified OBS path. Storage Path must be set to a specific directory in an OBS bucket rather than the root directory of the OBS bucket. EVS and SFS are all mounted to the <code>/home/ma-user/work</code> directory. You can add a data storage path during the runtime of a notebook instance by referring to Dynamically Mounting an OBS Parallel File System. Data stored in the directory is retained even if the notebook instance is stopped or restarted. When a notebook instance is deleted, the EVS storage is released and the stored data is not retained. SFS can be

Parameter		Description
		mounted to a new notebook instance and data can be retained.
Expansion Storage		This is a whitelist function. Submit a service ticket for a trial if needed. If you have multiple data storage paths, click Add PFS, Add OBS, or Add SFS as required to add specific storage mount directories. Constraints: • For each type, a maximum of five directories can be mounted. • The directories must be unique and cannot be mounted to a blacklisted directory. Nested mounting is allowed. Blacklisted directories are those with the following prefixes: /data/, /cache/, /dev/, /etc/, /bin/, /lib/, /sbin/, /modelarts/, /train-worker1-log/, /var/, /resource_info/, /usr/, /sys/, /run/, /tmp/, /infer/, and /opt/ After this parameter is configured, the notebook instance details page is displayed. Click Storage > Extended Storage to view or edit the extended storage information. If the number of storage devices does not reach the maximum, you can click Add Extended Storage.
Authentication Information	Secret	This parameter is mandatory when Storage Type is set to Object Storage Service or Parallel File System. Select an existing secret or click Create Credential on the right to create one. On the displayed DEW console, create a secret, and enter the secret key/value and user AK/SK.
More Settings	Remote SSH	After you enable this function, you can remotely access the development environment of the notebook instance from your local development environment. When a notebook instance is stopped, you can update the SSH configuration on the instance details page. The notebook instances with remote SSH enabled have VS Code plug-ins (such as Python and Jupyter) and the VS Code server package pre-installed, which occupy about 1 GB persistent storage space.

Parameter		Description
	Key Pair	Set a key pair after remote SSH is enabled. You can select an existing key pair or click Create Key Pair on the right to create one on the displayed Create Account Key Pair page. After a notebook instance is created, you can change the key pair on the instance details page. CAUTION Download the created key pair and properly keep it. When you use a local IDE to remotely access the notebook development environment, the key pair is required for authentication.
	Whitelist	Set a whitelist after remote SSH is enabled. This parameter is optional. Add the IP addresses for remotely accessing the notebook instance to the whitelist, for example, the IP address of your local PC or the public IP address of the source device. A maximum of five IP addresses can be added and separated by commas (,). If the parameter is left blank, all IP addresses will be allowed for remote SSH access. If your source device and ModelArts are isolated from each other in network, obtain the public IP address of your source device using a mainstream search engine, for example, by entering "IP address lookup", but not by running ipconfig or ifconfig/ip locally. After a notebook instance is created, you can change the whitelist IP addresses on the instance details page.
	Network	If this function is enabled, you can configure the VPC to connect the notebook instance to the network. This is a whitelist function. Submit a service ticket for a trial if required. After this function is enabled, the instance can be mounted to your VPC to implement access of multiple network planes. Before using this function, you need to configure VPC fine-grained access authorization by referring to Configuring Agency Authorization for ModelArts with One Click. If you have the VPC Administrator permission, you do not need to set this parameter. • VPC: Select an existing VPC from the drop-down list or create one as required. • Subnet: After you select a VPC, the default subnet is displayed. Select an existing subnet from the drop-down list or create one as required. • Security Group: Select an existing security group or create one as required.

4. After all parameters are set, the configuration summary is displayed on the right, and the fee is displayed in the lower part. Confirm the information and click **Next**.

The final bill may vary slightly. Switch to the notebook instance list. The notebook instance is being created. It will take several minutes before its status changes to **Running**. Then, the notebook instance is created.

5. In the notebook instance list, click the instance name to access its details page, and view the instance configurations.

If **Remote SSH** is enabled, you can click the modification icon on the right of the whitelist to modify it. You can click the modification icon on the right of **Authentication** to update the key pair of a stopped notebook instance.

In the **Storage** tab, click **Mount Storage** to mount an OBS parallel file system to the instance for reading data. For details, see [Dynamically Mounting an OBS Parallel File System](#).

If an EVS disk is used, click **Expansion** on the right of **Storage Capacity** to dynamically expand the EVS disk capacity. For details, see [Dynamically Expanding EVS Disk Capacity](#).

Accessing a Notebook Instance

Access a notebook instance in the **Running** state for coding.

- Accessed online using JupyterLab. For details, see [Using a Notebook Instance for AI Development Through JupyterLab](#).
- Remotely accessed from a local IDE through PyCharm. For details, see [Using Notebook Instances Remotely Through PyCharm](#).
- Remotely accessed from a local IDE through VS Code. For details, see [Using Notebook Instances Remotely Through VS Code](#).
- Remotely accessed from a local IDE through SSH. For details, see [Using a Notebook Instance Remotely with SSH](#).

ModelArts notebook instances are started by **ma-user** by default. After you access the instance, the default working directory is **/home/ma-user/work**.

Figure 5-7 Working directory example



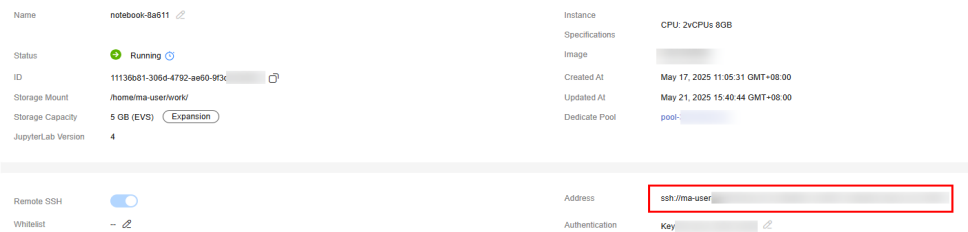
Most notebook instances in a dedicated resource pool are started as user **root**. The details are as follows:

- When you log in to the terminal as user **root**, the system automatically runs the **source /home/ma-user/.bashrc** command to synchronize the

environment variables of user **ma-user**. To disable this function, set the environment variable **export DISABLE_MA_USER_BASHRC to true** in the custom image to prevent the **/home/ma-user/.bashrc** file from being loaded.

- If the instance is started by user **root**, only user **root** can be used for SSH remote connection.

Figure 5-8 Using user **root** for SSH remote connection



JupyterLab 4.3.1

This version has significantly improved user experience, functions, and performance. The table below lists the main updates. For further information, see [JupyterLab Documentation](#).

Table 5-8 Feature updates

Function Module	Description
Workspaces	You can create multiple workspaces to organize and manage different projects and files, improving project management efficiency.
Launcher creation toolbar	Added a toolbar for functional access and convenient operations.
Theme framework	Optimized the theme framework to enhance the theme flexibility and compatibility. You can customize the GUI as required.
Configuration page	Optimized user experience on the configuration page for easier use.
Performance and debugging	Fully optimized the system performance and enhanced the debugging function to improve the development experience.
Code editor	Improved the code editor by adding more functions and optimizing performance to improve the code writing and editing experience.
Search function	Enhanced the search function for quicker and more accurate search.
Performance	Improved the overall performance, system response speed, and system stability.

Function Module	Description
Custom CSS style sheets	Custom CSS style sheets can be used to adjust the GUI appearance to meet personalized requirements.
Markdown charts	Charts can be used in Markdown to facilitate document writing.
Virtual scroll bar	Introduced the virtual scroll bar to improve the scrolling experience of large files or a large amount of content.
Workspace UI	Improved the workspace UI to provide better visual effect and operation experience and enhance UI friendliness.
File access records	You can view recently opened and closed files to quickly access common files and improve work efficiency.
Keyboard shortcuts	Keyboard shortcuts are improved for higher operation efficiency and convenience.

Mounting Directories of Notebook Containers

When you use EVS storage when creating a notebook instance, the **/home/ma-user/work** directory is used as the workspace for persistent storage.

The data stored in only the **work** directory is retained after the instance is stopped or restarted. When you use a development environment, store the data for persistence in **/home/ma-user/work**.

For details about mounting directories of a notebook instance, see [Table 5-9](#). The mounting points listed in the following table are not saved when images are saved.

Table 5-9 Mounting directories

Mount Point	Read Only	Remarks
/home/ma-user/work/	No	Persistent directory of your data.
/data	No	Mount directory of your PFS.
/cache	No	Used to mount the hard disk of the host NVMe (supported by bare metal specifications).
/train-worker1-log	No	Compatible with training job debugging.
/dev/shm	No	Used for PyTorch engine acceleration.

Selecting Storage for a Notebook Instance

Storage varies depending on performance, usability, and cost. No storage media can cover all scenarios. Learning about in-cloud storage application scenarios for better usage.

Table 5-10 On-cloud storage applications

Storage Type	Application	Advantage	Disadvantage
EVS	Data and algorithm exploration only in the development environment.	Block storage SSDs feature better overall I/O performance than NFS. The storage capacity can be dynamically expanded to up to 4096 GB. As persistent storage, EVS disks are mounted to /home/ma-user/work . The data in this directory is retained after the instance is stopped. The storage capacity can be expanded online based on demand.	This type of storage can only be used in a single development environment.

Storage Type	Application	Advantage	Disadvantage
PFS	NOTE - PFS is a whitelist function. To use this function, contact Huawei technical support. - Only OBS PFS in the same region can be mounted. PFS buckets mounted as persistent storage for AI development and exploration. - Storage for datasets. Datasets stored in the PFS buckets are directly mounted to notebooks for browsing and data processing and can be directly used during training. Select PFS when creating a notebook instance. After the instance is running, the OBS parallel file system that carries the datasets is dynamically mounted to the notebook instances. For details, see Dynamically Mounting an OBS Parallel File System. - Storage for code. After debugging on a notebook instance, specify the OBS path as the code path for starting training, facilitating	PFS is an optimized high-performance object storage file system with low storage costs and large throughput. It can quickly process high-performance computing (HPC) workloads. PFS mounting is recommended if OBS is used. NOTE Package or split the data to be uploaded by 128 MB or 64 MB. Download and decompress the data in local storage for better I/O and throughput performance.	Due to average performance in frequent read and write of small files, PFS storage is not suitable for large model training or file decompression. NOTE Before mounting PFS storage to a notebook instance, grant ModelArts with full read and write permissions on the PFS bucket. The policy will be retained even after the notebook instance is deleted.

Storage Type	Application	Advantage	Disadvantage
	temporary modification. 3. Storage for training check. Mount storage to the training output path such as the path to training logs. In this way, view and check training on the notebook instance in real time. This is especially suitable for analyzing the output of jobs trained using TensorBoard .		
OBS	NOTE • OBS is a whitelist function. To use this function, contact Huawei technical support. • Only OBS objects in the same region can be mounted. When uploading or downloading a large amount of data in the development environment, you can use OBS buckets to transfer data.	Low storage cost and high throughput, but average performance in reading and writing small files. It is a good practice to package or split the file by 128 MB or 64 MB. In this way, you can download the packages, decompress them, and use them locally.	The object storage semantics is different from the Posix semantics and needs to be further understood.

Storage Type	Application	Advantage	Disadvantage
SFS	Available only in dedicated resource pools. Use SFS storage in informal production scenarios such as exploration and experiments. One SFS device can be mounted to both a development environment and a training environment. In this way, you do not need to download data each time your training job starts. This type of storage is not suitable for heavy I/O training on more than 32 cards.	SFS is implemented as NFS and can be shared between multiple development environments and between development and training environments. This type of storage is preferred for non-heavy-duty distributed training jobs, especially for the ones not requiring to download data additionally when the training jobs start.	The performance of the SFS storage is not as good as that of the EVS storage.
OceanStor Pacific storage (SFS capacity-oriented 2.0)	Currently, it can be used only in Tiangong resource pools. It is suitable for the training jobs that use the file systems provided by SFS capacity-oriented 2.0 for AI model training and exploration. In addition, OBS APIs are provided to import training data from outside the cloud.	It provides a high-performance file client to meet the high storage bandwidth requirements of heavy-load training jobs. It also supports access to OBS. After the training data is imported to the storage via OBS APIs, it can be used to train the model directly without any conversion.	It provides the object storage semantics, which is different from the Posix semantics and needs to be further understood.

Storage Type	Application	Advantage	Disadvantage
Local storage	First choice for heavy-duty training jobs.	High-performance SSDs for the used VM or BMS, featuring high file I/O throughput. For heavy-duty training jobs, store data in the target directory and then start training. By default, the storage is mounted to the /cache directory. For details about the available space of the /cache directory, see What Are Sizes of the /cache Directories for Different Notebook Specifications in DevEnviron?	The storage lifecycle is associated with the container lifecycle. Data needs to be downloaded each time the training job starts.

FAQ

- How do I use EVS in a development environment?
[When creating a notebook instance](#), select a small-capacity EVS disk. You can scale out the disk as needed. For details, see [Dynamically Expanding EVS Disk Capacity](#).
- How do I use an OBS parallel file system in a development environment?
When training data in a notebook instance, you can use the datasets mounted to a notebook container, and use an OBS parallel file system. For details, see [Dynamically Mounting an OBS Parallel File System](#).
- How do I switch back to JupyterLab 3.2.3 if an error occurs during the startup of JupyterLab 4.3.1?
Locate the target instance in the list and click **Start** in the **Operation** column. In the displayed dialog box, select JupyterLab 3.2.3 and click **OK**.
- Can I use both JupyterLab 3.2.3 and 4.3.1 in a project?
Not recommended. Each JupyterLab instance runs independently. Therefore, you need to create an instance for each version. To try different versions, you can start them in different containers or environments. Pay attention to the following:
 - The configuration file and data path may vary according to the version. Ensure the independence of data and configuration.
 - Running multiple versions at the same time may cause port conflicts or other resource competition problems.
- Can I use GDB in a notebook instance?
No. GDB needs Docker with privileged containers. For security purposes, the development environment does not allow privileged containers. Therefore, GDB cannot be used in notebook instances.

5.4 Managing Notebook Instances

5.4.1 Searching for a Notebook Instance

Searching for an Instance

All created instances are displayed on the notebook page. To display a specific instance, search for it based on filter criteria.

- **Grant the permission to the IAM user for viewing all notebook instances.** Log in to the ModelArts management console. In the navigation pane on the left, choose **Development Workspace > Notebook**. On the displayed page, enable **View all**.
- Set search criteria, such as name, ID, status, image, flavor, description, and creation time.

Assigning the Required Permissions

Any IAM user granted with the **listAllNotebooks** and **listUsers** permissions can click **View all** on the notebook page to view the instances of all IAM users in the current IAM project. After the permission is granted, you can access OBS and SWR of IAM users in a notebook instance.

1. Log in to the ModelArts management console as a tenant user, hover the cursor over your username in the upper right corner, and choose **Identity and Access Management** from the drop-down list to switch to the IAM management console.
2. On the IAM console, choose **Permissions > Policies/Roles** from the navigation pane, click **Create Custom Policy** in the upper right corner, and create two policies.

Policy 1: Create a policy that allows users to view all notebook instances of an IAM project, as shown in [Figure 5-9](#).

- **Policy Name:** Enter a custom policy name, for example, **Viewing all notebook instances**.
- **Policy View:** Select **Visual editor**.
- **Policy Content:** Select **Allow**, **ModelArts Service**, **modelarts:notebook:listAllNotebooks**, and default resources.

Figure 5-9 Creating a custom policy

The screenshot shows the 'Create Custom Policy' interface in the IAM console. It includes fields for Policy Name, Policy View (Visual editor), Policy Content (Allow, ModelArts Service, modelarts:notebook:listAllNotebooks), Description, and Scope (Project-level services). The OK button is highlighted.

Policy 2: Create a policy that allows users to view all users of an IAM project.

- **Policy Name:** Enter a custom policy name, for example, **Viewing all users of the current IAM project**.
- **Policy View:** Select **Visual editor**.
- **Policy Content:** Select **Allow, Identity and Access Management**, **iam:users:listUsers**, and default resources.

3. In the navigation pane, choose **User Groups**. Then, click **Authorize** in the **Operation** column of the target user group. On the **Authorize User Group** page, select the custom policies created in 2, and click **Next**. Then, select the scope and click **OK**.

After the configuration, all users in the user group have the permission to view all notebook instances created by users in the user group.

If no user group is available, create a user group, add users using the user group management function, and configure authorization. If the target user is not in a user group, you can add the user to a user group through the user group management function.

Starting Notebook Instances of Other IAM Users

If an IAM user wants to access another IAM user's notebook instance through remote SSH, they need to update the SSH key pair to their own. Otherwise, error **ModelArts.6786** will be reported. For details about how to update a key pair, see [Modifying the SSH Configuration for a Notebook Instance](#). The error message is as follows: "ModelArts.6789: Failed to find SSH key pair xxx on the ECS key pair page. Update the key pair and try again later."

5.4.2 Updating a Notebook Instance

Changing an Image

ModelArts allows you to change images on a notebook instance to flexibly adjust its AI engine. Images can be changed for only stopped notebook instances.

CAUTION

After the image is changed, the notebook instance may fail to be started, as the notebook instance specifications corresponding to the image do not match. The charging rule may also change. Exercise caution when performing this operation.

1. Log in to the ModelArts console. In the navigation pane on the left, choose **Development Workspace > Notebook**.
2. Locate the target notebook instance in the list and choose **More > Change Image** in the **Operation** column.
3. In the displayed window, select a new image, confirm the operation, and click **OK**.

The images will be matched based on the hardware model. If the images do not match, they cannot be selected.

After the image is changed, you can view it in the **Image** column on the notebook page.

Changing the Specifications of a Notebook Instance

ModelArts allows you to change the node specifications for a notebook instance. Specifications of a notebook instance can be modified only when the notebook instance is in the **Stopped**, **Running**, or **Startup failed** state.

1. Log in to the ModelArts console. In the navigation pane on the left, choose **Development Workspace > Notebook**.
2. Locate the target notebook instance in the list and choose **More > Modify Instance Specifications** in the **Operation** column.
3. In the displaying dialog box, select the target specifications, confirm the configuration, and click **OK**.

After the specifications are changed, you can view it in the **Instance Specifications** column on the notebook page.

Figure 5-10 Modify specifications

Change Specifications

^ Resource Change Details

The specifications of the following notebook instance will be changed.

Name	Status	Image	Instance S...	Created At	Created By
notebook-...	Running	pytorch1.4...	CPU: 2vC...	Mar 11, 20...	ei_modelar...

Current Specifications

CPU: 2vCPUs 8GB

Target Specifications

Select specifications.

Confirm Configurations (Changed Resources: 1)

Current Specifications

Specifications

CPU: 2vCPUs 8GB

New Specifications

Specifications

Select specifications.

Confirm

☐ I acknowledge the price change and agree to proceed.

OK

NOTE

The instance specifications can be changed only when there are other options in the cluster. If no other specifications are available, the change cannot be performed.

Modifying the SSH Configuration for a Notebook Instance

ModelArts allows you to modify the SSH configuration only when the notebook instance is stopped.

If remote SSH connection is not configured when you create the notebook instance and you want to enable it after the instance is created, go to the notebook instance details page, and enable SSH configuration. If a whitelist is configured for remote connection and you need to use another IP address, modify the IP addresses in the whitelist on the instance details page. If an IAM user needs to start a notebook instance of another user's, change the key pair.

1. Log in to the ModelArts console. In the navigation pane on the left, choose **Development Workspace > Notebook**.
2. Click the target notebook instance. Enable remote SSH and change the key pair and whitelist.

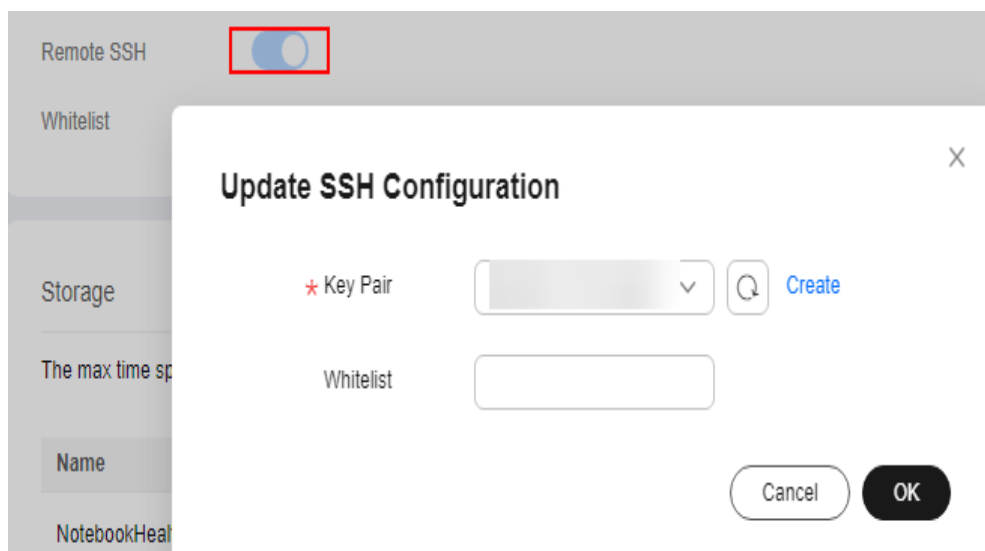
NOTE

For manually enabled remote SSH, see [Figure 5-11](#). After the SSH configuration is updated, the remote SSH function cannot be disabled.

If SSH remote development must be configured for the selected image, configure

Whitelist or click  next to **Authentication**.

Figure 5-11 Updating SSH configuration



- Click ▼ and choose an existing key pair, or click **Create** to create one.
- After you change the IP addresses, the existing links are still valid. After the links are released, the new links only from the changed IP addresses can be set up.
- Ensure that public IP addresses are set. If your source device and the Huawei Cloud ModelArts are isolated from each other in network, obtain the public IP address of your source device using a mainstream search engine, for example, by entering "IP address lookup", but not by running **ipconfig** or **ifconfig/ip** locally.

5.4.3 Starting, Stopping, or Deleting a Notebook Instance

Starting or Stopping an Instance

Stop the notebook instances that are not needed. You can also restart a stopped instance.

1. Log in to the ModelArts console. In the navigation pane on the left, choose **Development Workspace > Notebook**.
2. Start or stop the target notebook instance.
 - To start a notebook instance, click **Start** in the **Operation** column of the target notebook instance. Only stopped notebook instances can be started.
 - To stop a notebook instance, click **Stop** in the **Operation** column of the target notebook instance. Only running notebook instances can be stopped.

CAUTION

After a notebook instance is stopped:

- Data in the `/home/ma-user/work` directory and directories dynamically mounted to `/data` is saved. Data in other directories will be deleted. For example, the external dependency packages installed in other directories in the development environment will be deleted. Save your development environment settings as an image. For details, see [Saving a Notebook Instance](#).
 - The notebook instance will no longer be billed. If the instance has an EVS disk attached, the storage space will still be billed.
-

Deleting an Instance

Delete the notebook instances that are not needed.

1. Log in to the ModelArts console. In the navigation pane on the left, choose **Development Workspace > Notebook**.
2. In the notebook list, locate the target notebook instance, and click **Delete** in the **Operation** column. In the displayed dialog box, confirm the information, enter **DELETE** in the text box, and click **OK**.

CAUTION

Deleted notebook instances cannot be recovered. After a notebook instance is deleted, the data stored in the mounted directory will be deleted.

5.4.4 Saving a Notebook Instance

To save a notebook environment image, do as follows: Create a notebook instance using a preset image, install custom software and dependencies on the base

image, and save the running instance as a container image. After the image is saved, the default working directory is the / path in the root directory.

In the saved image, the installed dependencies are retained. The data stored in **home/ma-user/work** for persistent storage will not be stored. When you use VS Code for remote development, the plug-ins installed on the Server are retained.

NOTE

- If the image fails to be saved, view the event on the notebook instance details page. For details, see [Viewing Notebook Events](#).
- The image to be saved should not be larger than 35 GB and there should be no more than 125 layers. Otherwise, the image may fail to be saved. For details, see [Space Allocation for Container Engines](#).
 - If a dedicated resource pool is used, log in to the ModelArts console. In the navigation pane on the left, choose **AI Dedicated Resource Pools > Elastic Clusters**. On the displayed page, configure the container engine size as needed. For details, see [Resizing a Dedicated Resource Pool](#).
 - If the fault persists, contact technical support.
- When you use the containerd image saving function, pay attention to the following:
 - Do not download files and save images at the same in an instance to prevent performance deterioration caused by resource competition.
 - If there is a file larger than 10 GB in an instance system folder (for example, **/opt/**), you should not use the image saving function, in case the image size is too large or the saving duration is too long.

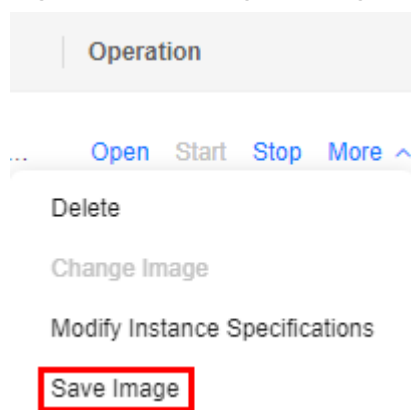
Prerequisites

The notebook instance is in **Running** state.

Saving an Image

1. Locate the target notebook instance in the list and choose **More > Save Image** in the **Operation** column.

Figure 5-12 Saving an image



2. In the displayed dialog box, configure the organization, image name, image version, and description, and click **OK**.

Choose an organization from the **Organization** drop-down list. If no organization is available, click **Create** to create one.

- Users in an organization can share all images in the organization.
- The image will be saved as a snapshot, and it will take about 5 minutes. During this period of time, do not perform any operations on the instance.

Figure 5-13 Saving as a snapshot



NOTICE

The time required for saving an image as a snapshot will be counted in the instance running duration. If the instance running duration expires before the snapshot is saved, saving the image will fail.

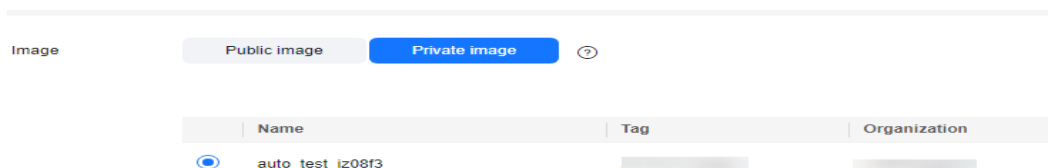
- After the image is saved, the instance status changes to **Running**. You can view the image details on the image management page.
- Click the name of the image to view its details.

Using a Custom Image to Create a Notebook Instance

The images saved from a notebook instance can be viewed on the **Image Management** page. You can use these images to create new notebook instances, which inherit the software configurations of the original notebook instances.

Method 1: On the **Create Notebook** page, click **Private Image** and select the saved image.

Figure 5-14 Selecting a custom image to create a notebook instance



Method 2: On the **Image Management** page, click the target image to access its details page. Then, click **Create Notebook**.

Which Data Can Be Saved When I Save an Image?

- Data that can be saved: Files and directories that are statically added to images during container building, for example, dependencies and the **/home/ma-user** directory are saved in the image environment.
- Data that cannot be saved: Mounting directories or data volumes that are dynamically connected to the host during container startup. You can run the **df -h** command to view the mounted dynamic directories. Data that is not in the **/** path will not be saved.

For example, data that is persistently stored in **home/ma-user/work** and data that is dynamically mounted to **/data** is not saved.

5.4.5 Dynamically Expanding EVS Disk Capacity

Overview

If a notebook instance uses an EVS disk for storage, the disk is mounted to **/home/ma-user/work/** of the notebook container and the disk capacity can be expanded by up to 100 GB at a time when the instance is running.

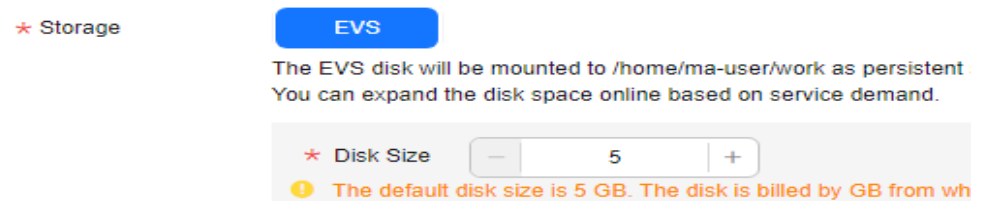
Application Scenarios

During notebook development, select a small EVS disk capacity, for example, 5 GB, when creating a notebook instance because the storage requirements are low at the initial stage. After the development, a large volume of data must be trained. Then, expand the disk capacity to cost-effectively meet your service needs.

Restrictions

- The target notebook instance must use EVS for storage.

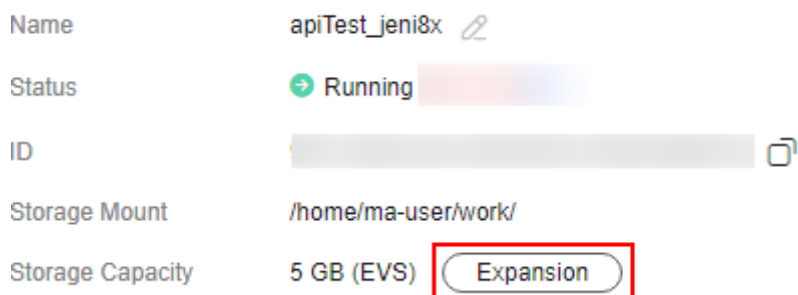
Figure 5-15 Selecting EVS when creating a notebook instance



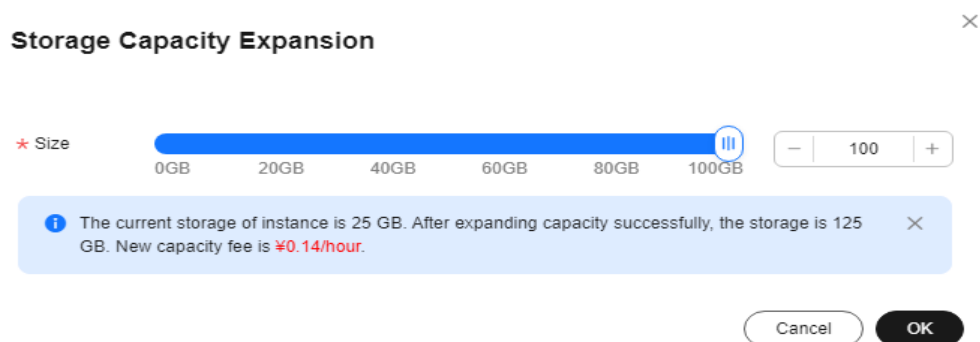
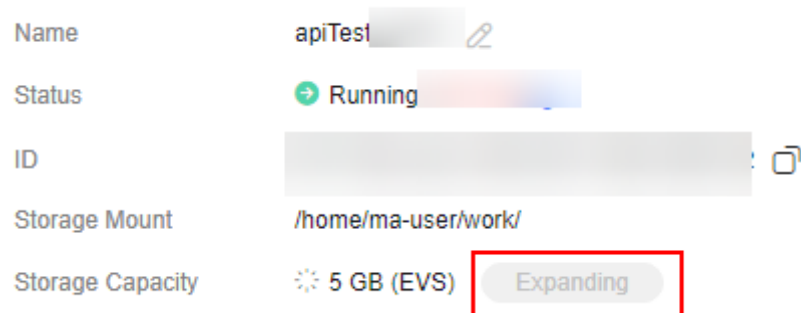
- Up to 100 GB can be expanded at a time. Additionally, the total capacity after expansion cannot exceed 4,096 GB.
- If the original capacity of an EVS disk is 4,096 GB, the disk capacity cannot be expanded.
- After the instance is stopped, the expanded capacity still takes effect. The billing is based on the expanded EVS disk capacity.
- An EVS disk is billed as long as it is used. To stop billing an EVS disk, delete data from the EVS disk and release the disk.

Procedure

1. Log in to the ModelArts management console. In the navigation pane on the left, choose **Development Workspace > Notebook**.
2. Click the name of a running notebook instance. On the instance details page, click **Expansion**.

Figure 5-16 Instance details page

- Set the capacity to be expanded and click **OK**. **Expanding** shows that the capacity expansion is in progress. After the expansion, the displayed storage capacity is the expanded capacity.

Figure 5-17 Capacity expansion**Figure 5-18** Expanding

5.4.6 Dynamically Mounting an OBS Parallel File System

Overview

A parallel file system is an optimized high-performance file system provided by Object Storage Service (OBS). For details, see [About Parallel File System](#).

Dynamic OBS mounting uses a mounting tool to convert the object storage protocol into the POSIX file protocol. OBS storage is simulated as a local file system and dynamically mounted to a running notebook container in ModelArts.

After the mounting, you can perform application operations on the OBS objects in the notebook container.

Application Scenarios

Scenario 1: After you mount the OBS storage in which the target dataset is stored to your notebook instance, you can preview and perform operations in the dataset like operating a local file system.

Scenario 2: When training data in a notebook instance, you can use the dataset mounted to a notebook container.

Restrictions

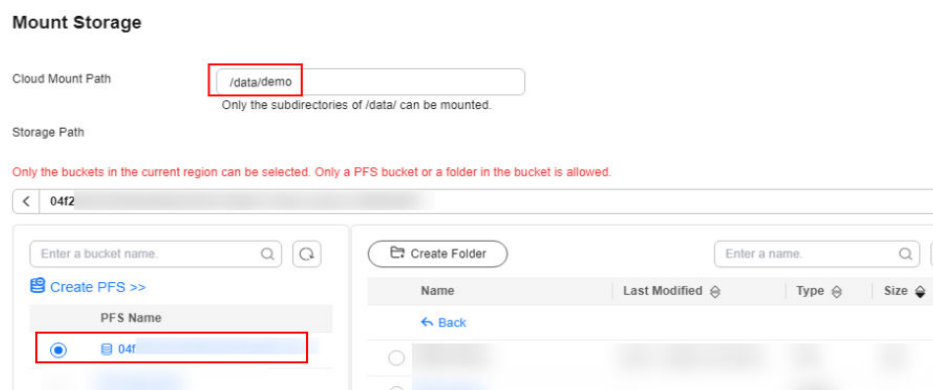
- OBS provides object buckets and parallel file systems for storage. The ModelArts notebook supports only the mounting of an OBS parallel file system to **/data/** of a notebook container.
- On the international website, dynamic mounting of OBS parallel file systems is supported only in CN-Hong Kong, AP-Bangkok, and AP-Singapore.

Procedure

Method 1: Through the ModelArts management console

1. Log in to the ModelArts management console. In the navigation pane on the left, choose **Development Workspace > Notebook**.
2. Select a running notebook instance and click its name. On the notebook instance details page, click the **Storage** tab. From there, click **Mount Storage** and configure mounting parameters.
 - a. Set a local mounting directory. Enter a folder name in **/data/**, for example, **demo**. The system will automatically create the folder in **/data/** of the notebook container to mount the OBS file system.
 - b. Select the folder for storing the OBS parallel file system and click **OK**.

Figure 5-19 Dynamically mounting an OBS parallel file system



3. View the mounting result on the notebook instance details page.

Figure 5-20 Successful mounting

Type	Status	Storage Path	Cloud Mount Path	Operation
Parallel File System	Mounted	obs://	/data/demo/	Unmount Storage

5.4.7 Viewing Notebook Events

Instance statuses and key operations such as creating, starting, and stopping an instance, and changing the instance flavor are recorded in the backend. You can view the events on the notebook instance details page to monitor the instance statuses. You can refresh events on the right of the **Event** tab. You can also set the interval for automatically refreshing events to 30 seconds, 1 minute, or 5 minutes.

Viewing Events of a Notebook Instance

To view the event details of a notebook, click the notebook name. On the displayed notebook details page, click the **Event** tab.

Notebook Instance Events

Table 5-11 Events during instance creation

Event	Description	Severity
Scheduled	The instance has been scheduled.	Warning
PullingImage	The image is being pulled.	Warning
PulledImage	The image has been pulled.	Warning
NotebookHealthy	The instance is running and healthy.	Major
CreateNotebookFailed	Creating an instance failed.	Critical
PullImageFailed	Pulling the image failed.	Critical
FailedCreate	Failed to create notebook container. Please contact SRE to check node {node_name}	Critical
CreateContainerError	Failed to create container. Please contact SRE to check node {node_name}	Critical
FailedAttachVolume	Failed to attach volume. Please contact SRE to check node {node_name}	Major

Event	Description	Severity
MountVolumeFailed	Mount volume failed; Check whether the DEW secret is correct if the instance cannot change to running in five minutes	Critical
	Mount volume failed; Check if vpc of sfs-turbo is interconnected if the instance cannot change to running in five minutes	Critical
	Mount volume failed; Please contact SRE to check node {node_name} if the instance cannot change to running in five minutes	Critical

Table 5-12 Events during instance stopping

Event	Description	Severity
StopNotebook	The instance has been stopped.	Major
StopNotebookResourceIdle	The notebook instance will automatically stop or has automatically stopped because resources are idle.	Major

Table 5-13 Events during instance update

Event	Description	Severity
UpdateName	Updating the instance name	Warning
UpdateDescription	Updating the instance description	Warning
UpdateFlavor	Updating the instance flavor	Major
UpdateImage	Updating the instance image	Major
UpdateStorageSize	The instance storage size is being updated. (User %s is updating storage size from %s GB to %s GB.)	Major

Event	Description	Severity
	The instance storage size has been updated. (User %s updated the storage size.)	Major
UpdateKeyPair	Configured the instance key pair. (User %s updated the instance key pair to { %s}.)	Major
	Updating the instance key pair (User %s updated the instance key pair from %s to %s.)	Major
UpdateWhitelist	Updating the instance access whitelist	Major
UpdateHook	Updating a custom script	Major
UpdateStorageSizeFailed	Updating the storage size failed because the resources are sold out. (EVS disks are sold out.)	Critical
	Updating the storage size failed due to an internal error. (Updating the EVS disk size failed. The O&M engineers are handling the fault.)	Critical

Table 5-14 Events during image saving

Event	Description	Severity
SaveImage	The image has been saved.	Major
SavedImageFailed	Saving the image failed due to processes in D status. (There are processes in 'D' status. Check process status using 'ps -aux' and kill all the processes in 'D' status.)	Critical

Event	Description	Severity
	Saving the image failed because the image is too large. (The container size (%dG) is greater than the threshold (%dG).)	Critical
	Saving the image failed due to the limit on the number of layers. (There are too many layers in your image.)	Critical
	Saving the image failed due to task timeout. (The O&M engineers are handling the fault.)	Critical
	Saving the image failed due to SWR service issues.	Critical
CheckImageSize	The notebook container image size is {image_size}G. {image_size} indicates the image size, which is a variable.	Warning
CheckImageLayer	The number of original notebook image layers is {layer_number}. {layer_number} indicates the number of image layers, which is a variable.	Warning
ContainerCommitStarted	Start to commit notebook container.	Warning
ContainerCommitSuccess	Notebook container commit successfully.	Warning
ImagePushStarted	Start to push notebook image.	Warning
ImagePushSuccess	Notebook image push successfully.	Warning

Event	Description	Severity
ContainerCommitFailed	Failed to commit notebook container. Please contact SRE to check node {node_name}. {node_name} indicates the node name, which is a variable and is generally in the format of an IP address, for example, 192.168.225.161 .	Warning
ImagePushFailed	Failed to push Notebook image. Please contact SRE to check node {node_name}.	Warning

Table 5-15 Events during instance running

Event Name	Description	Severity
NotebookUnhealthy	The instance is unhealthy.	Critical
OutOfMemory	The instance is out of memory.	Critical
JupyterProcessKilled	The Jupyter process has been stopped.	Critical
CacheVolumeExceed-Quota	The /cache file size has exceeded the upper limit.	Critical
NotebookHealthy	The instance has been restored to the healthy state.	Major
EVSSoldOut	EVS disks are sold out.	Critical

Table 5-16 Events for dynamic OBS mounting

Event	Description	Severity
DynamicMountStorage	The OBS storage is mounted.	Major
DynamicUnmountStorage	The OBS storage is unmounted.	Major

Table 5-17 Events triggered on the user side

Event	Description	Severity
RefreshCredentialsFailed	Authentication failed.	Critical

5.4.8 Notebook Cache Directory Alarm Reporting

When creating a notebook instance, you can select CPU, GPU, or Ascend resources based on the service data volume. If you select GPU or Ascend resources, ModelArts mounts hard disks to the cache directory. You can use this directory to store temporary files.

Capacity alarms are not generated for the cache directory of the notebook instance by default. Exceeding the capacity limit will restart the notebook instance. After the restart, multiple configurations are reset, discarding your data and losing the environment. This will affect your experience. You are advised to enable the monitoring and alarms for the cache directory usage and report the data to AOM.

Configuration Process

1. Enter the basic alarm information.
2. [Set an alarm rule.](#)
 - a. Configure monitoring metrics.
 - b. Set alarm triggering conditions.
3. [Configure alarm notifications.](#)
 - a. Create a topic, configure the topic policy, and subscribe to the topic.
 - b. Create an alarm action rule.
 - c. Select the created action rule.

Configuring Alarm Settings

1. Log in to the AOM console.
2. Choose **Alarm Center** > **Alarm Rules** and click **Create Alarm Rule**.
3. Enter the basic alarm information.

Basic Information

* Rule Name

Description

0/1,024

4. Set an alarm rule.

Rule Type: Select **Threshold alarm**.

Monitored Object: Select **Select resource objects**. Click **Select Resource Object**. A new dialog box is displayed.

- **Add By:** Select **Dimension**.
- **Metric Name:** Click **Custom Metrics** and select the cache metrics to be monitored. Example: **ma_container_notebook_cache_dir_size_bytes** (total size of the cache directory) and **ma_container_notebook_cache_dir_util** (usage of the cache directory)
- **Dimension:** Select a metric dimension, for example, *service_id:xxx*, and click **Confirm**.

After setting the monitored object, set **Statistic** and **Statistical Period**.

Alarm Condition: Set this parameter based on your needs.

Figure 5-21 Select Monitored Object

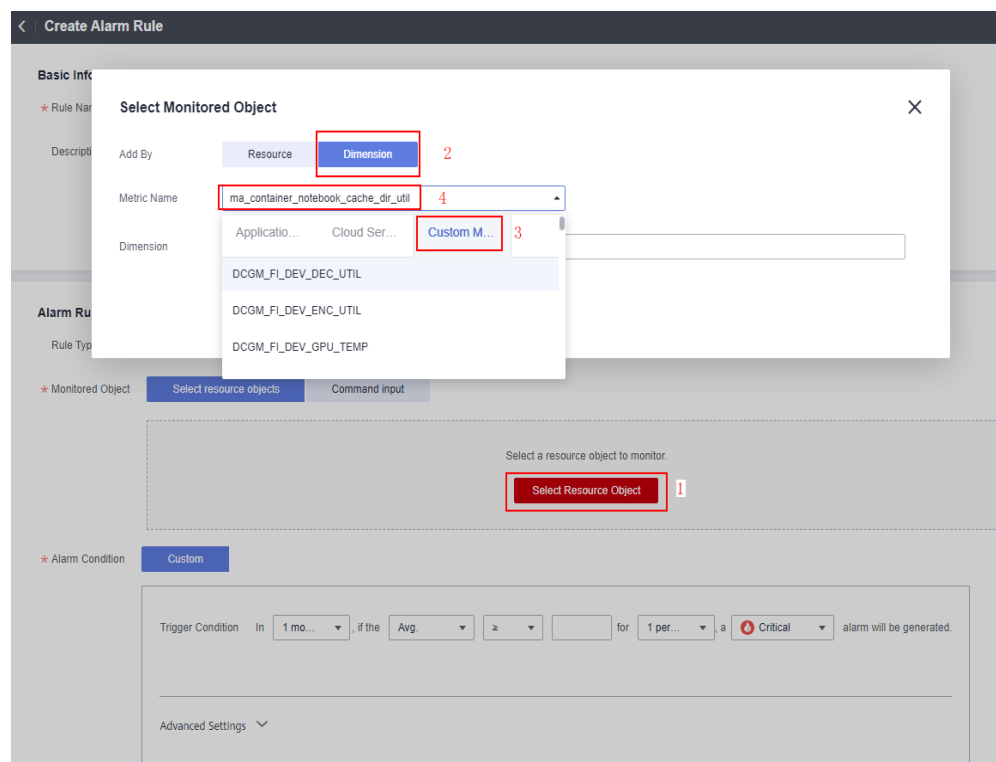


Figure 5-22 Configuring statistics method

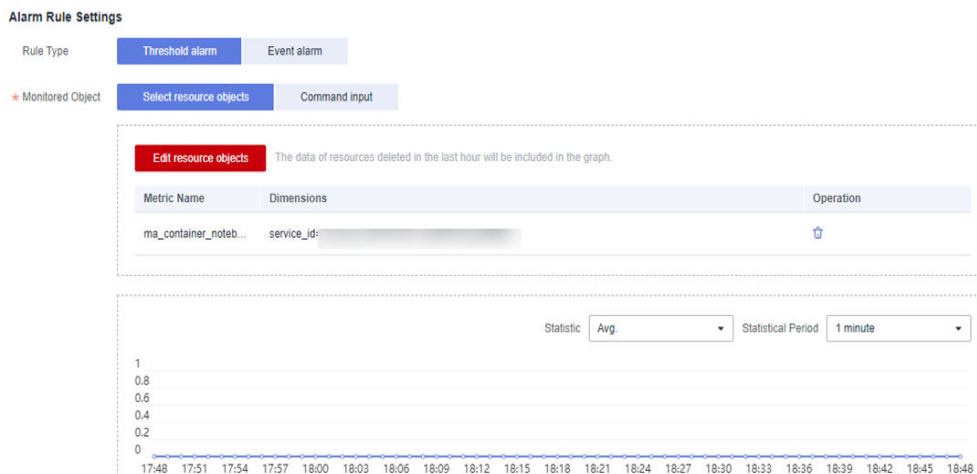
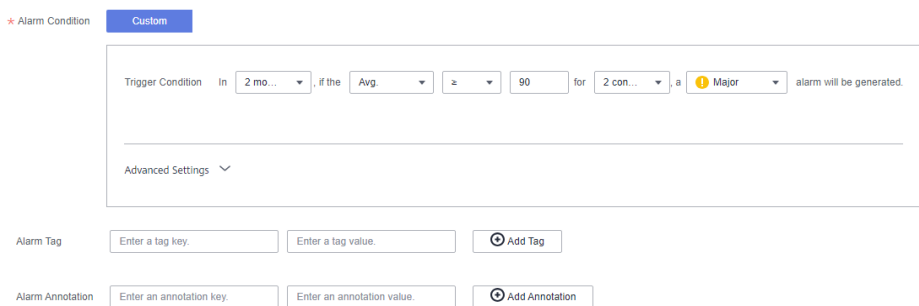
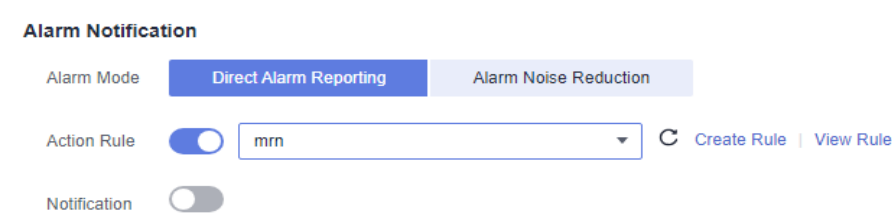


Figure 5-23 Configuring alarm conditions



5. Configure alarm notifications and click **Create Now**.
- Alarm Mode:** Select **Direct Alarm Reporting**.
- Action Rule:** Enable it and select the created action rule. If the existing alarm action rules cannot meet your requirements, click **Create Rule** to create an action rule. For details, see [Creating an Alarm Action Rule](#).
- Notification:** Enable it.

Figure 5-24 Configuring alarm notifications



- Create a topic in SMN to configure alarm notification rules.
- **Creating a Topic**
 - i. Go to the SMN console. In the navigation pane, choose **Topic Management > Topics**.

- ii. Click **Create Topic**. Enter a topic name, select an enterprise project, and click **OK**.
- iii. Locate the target topic and choose **More > Configure Topic Policy** in the **Operation** column.
Select **APM** to allow AOM alarms to trigger SMN.

Figure 5-25 Configure Topic Policy

Configure Topic Policy

Topic Name test

Policy [?](#) **Basic**

Users who can publish messages to this topic

☒ Topic creator

☐ All users

☐ Specified user accounts

Enter one or more account IDs or URNs, each on a separate line.

[Learn how](#) to obtain an account ID.

Services that can publish messages to this topic

☐ OBS ☐ DWS ☒ APM ☐ AAD ☐ EFS ☐ VOD

☐ MPC ☐ LTS ☐ CTS

OK Cancel

- iv. Click **Add Subscription** in the **Operation** column of the topic. After the subscription is successful, a notification is received once the alarm conditions are met.
Select a protocol, such as email or SMS, and enter the endpoints, such as email addresses or mobile numbers. Click **OK**.

Add Subscription

Basic Information

Topic Name mrn_test

* Protocol **Email**

* Endpoint [?](#) Endpoints Description

[+](#) Add Endpoint

[Batch Add Endpoints](#)

A record is displayed in the subscription list, but the record is in the **Unconfirmed** state.

Subscription URN	Protocol	Endpoint	Request Header	Description	Topic Name	Status	Operation
urn:smn:ap-southeast-1:160...	SMS		-	-	test	Unconfirmed	Request Confirmation Delete

After receiving the email, confirm the subscription.

Then, the subscription is in the confirmed state.

- **Creating an Alarm Action Rule**
An action rule specifies how AOM notifies you when an alarm is triggered. After an alarm action rule is enabled, the system sends notifications based on the associated SMN topic and message template. Enter the action rule name, select the action rule type, select the topic created in [the previous step](#), select a message template, and click **Confirm**.

Figure 5-26 Create Alarm Action Rule

Create Alarm Action Rule

★ Rule Name

Enter an action rule name.

Enter 1 to 100 characters and do not start or end with an underscore (_) or hyphen (-). Only letters, digits, underscores, and hyphens are allowed.

Description

Enter a description.

0/1,024

Enter up to 1,024 characters. Only letters, digits, space, and special characters (_ *) are allowed. Do not start or end with an underscore (_).

★ Action Type

Notification

★ Topic

-Select-

C

If you do not see a topic you like, create one on the SMN console.

★ Message Template

aom.built-in.template.en

C

Create Template | View Template

Confirm

Cancel

In the **Alarm Notification** area of the **Create Alarm Rule** page, set **Action Rule** to the newly created alarm action rule and click **Create Now**.

After the configuration is complete, you will receive an email notification once the alarm conditions are met.

5.5 Using a Notebook Instance for AI Development Through JupyterLab

5.5.1 Using JupyterLab to Develop and Debug Code Online

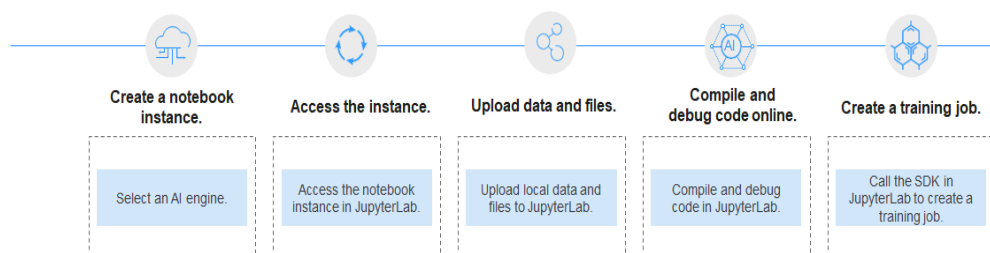
JupyterLab is an interactive development environment, enabling you to compile notebooks, operate terminals, edit Markdown text, enable interaction, and view

CSV files and images. JupyterLab is the future mainstream development environment for developers.

ModelArts allows you to access notebook instances online using JupyterLab and develop AI models based on the PyTorch, TensorFlow, or MindSpore engines.

Figure 5-27 shows the operation process.

Figure 5-27 Using JupyterLab to develop and debug code online



Procedure

1. Create a notebook instance.

On the ModelArts management console, create a notebook instance with a proper AI engine. For details, see [Creating a Notebook Instance \(Default Page\)](#).

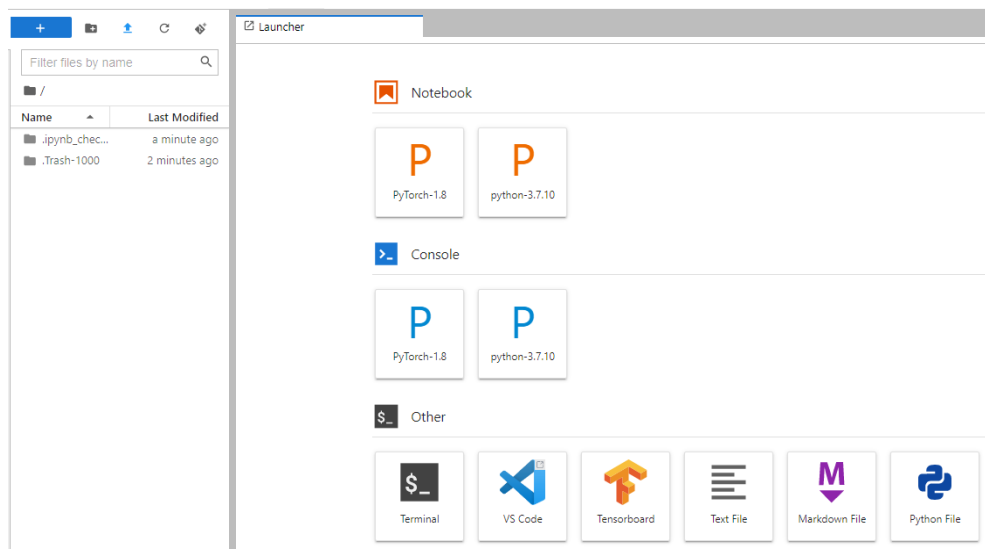
2. After the notebook instance is created, it is in the **Running** state. Locate it in the list and click **Open** in the **Operation** column to access JupyterLab.

Figure 5-28 Accessing a notebook instance

Name	Status	Image	Flavor	Description	Created At	Created By	Operation
notebook-779 40769745-745c-439c-8f7b-6729a2a77360	Running (58 minutes left)	sparc3.1.1-ubuntu18.04	CPU: 2vCPUx86GB	--	Mar 20, 2024 11:16:32 AM		Open Start Stop More

3. The **Launcher** page is automatically displayed. Perform required operations. For details, see [JupyterLab Documentation](#).

Figure 5-29 JupyterLab homepage

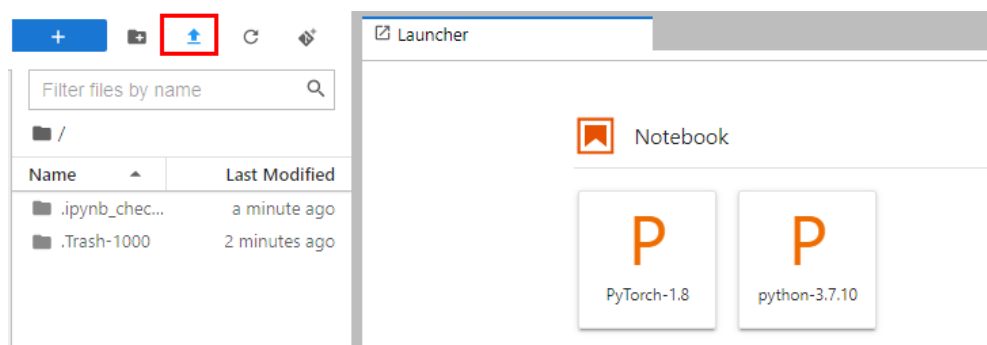


NOTE

The notebook and console kernels and versions displayed on the **Launcher** page vary depending on the AI engine based on which a notebook instance is created. [Figure 5-29](#) shows an example only. Obtain the notebook and console kernels and versions on the management console.

4. Upload training data and code files to JupyterLab. For details, see [Uploading Files from a Local Path to JupyterLab](#).

Figure 5-30 Button for uploading a file

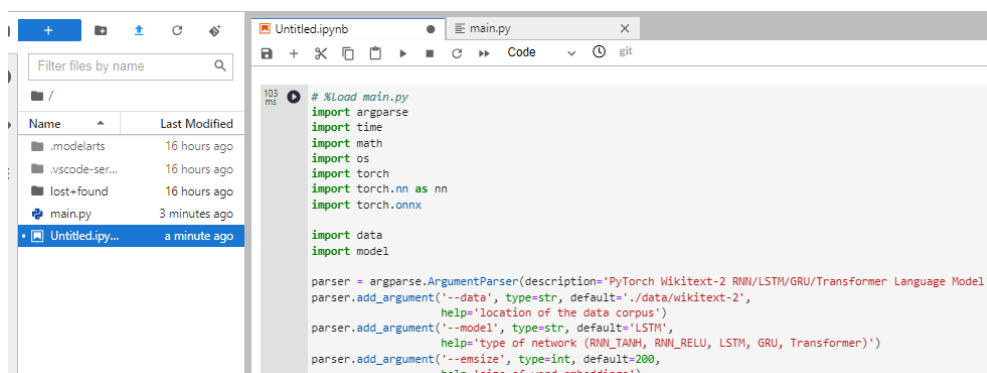


5. In the navigation pane on the left, double-click the uploaded code file, compile the file in JupyterLab, and debug it. For details about how to use JupyterLab, see [Common Functions of JupyterLab](#).

NOTE

If your code file is in .py format, open a new .ipynb file and run the `%load main.py` command to load the content of the .py file to the .ipynb file for encoding and debugging.

Figure 5-31 Opening a code file



6. In JupyterLab, call the ModelArts SDK to create a training job for in-cloud training.
For details, see [Creating a Training Job](#).

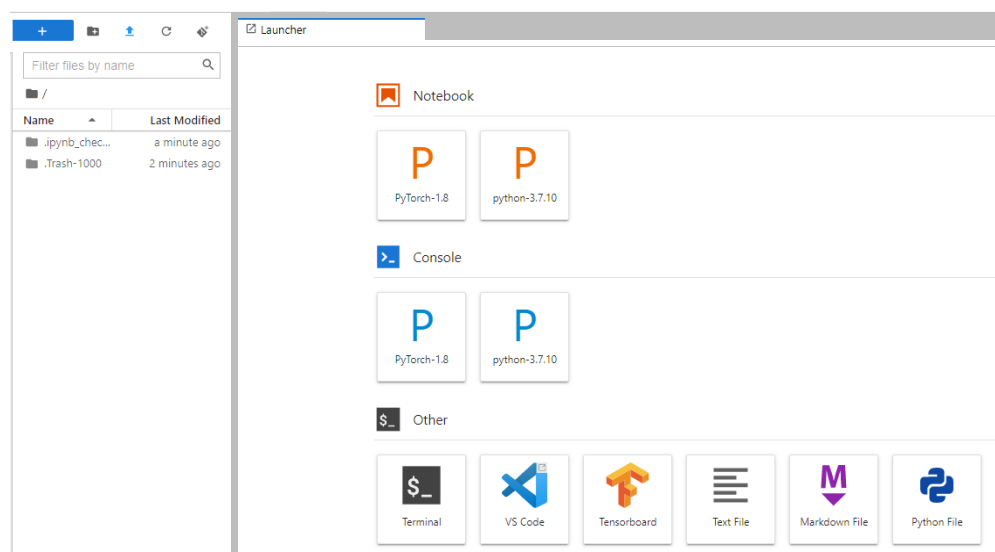
5.5.2 Common Functions of JupyterLab

Introduction

To access JupyterLab from a running notebook instance, perform the following operations:

1. Log in to the ModelArts management console. In the navigation pane on the left, choose **Development Workspace > Notebook**.
2. Click **Open** in the **Operation** column of a running notebook instance to access JupyterLab.
3. The **Launcher** page is automatically displayed. Perform required operations. For details, see [JupyterLab Documentation](#).

Figure 5-32 JupyterLab homepage



NOTE

The notebook and console kernels and versions displayed on the **Launcher** page vary depending on the AI engine based on which a notebook instance is created. [Figure 5-32](#) shows an example only. Obtain the notebook and console kernels and versions on the management console.

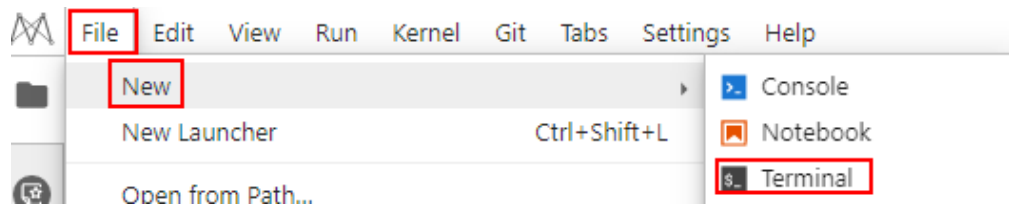
- **Notebook:** Select a kernel for running notebook, for example, TensorFlow or Python.
- **Console:** Call the terminal for command control.
- **Other:** Edit other files.

Creating a Terminal in JupyterLab

You can run Python commands on the terminal to operate the terminal. The following describes how to open the terminal of JupyterLab.

1. Create a notebook instance. When the instance is running, click **Open** in the **Operation** column. The **JupyterLab** page is displayed.
2. Choose **File > New > Terminal**. The **Terminal** page is displayed.

Figure 5-33 Terminal



3. You can use **pip** to install external libraries in the **TensorFlow-1.8** environment on the **Terminal** page. For example, to install Shapely:
- Enter the following commands in the code input box to obtain the kernel of the current environment and activate the Python environment on which the installation depends:

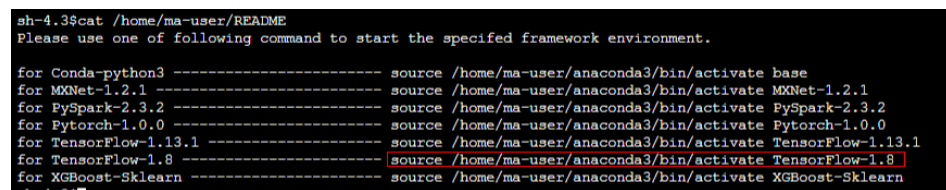
```
cat /home/ma-user/README
```

```
source /home/ma-user/anaconda3/bin/activate TensorFlow-1.8
```

NOTE

To install TensorFlow in another Python environment, replace **TensorFlow-1.8** in the command with the target engine.

Figure 5-34 Activating the environment



Run the following command in the code input box to install Shapely:

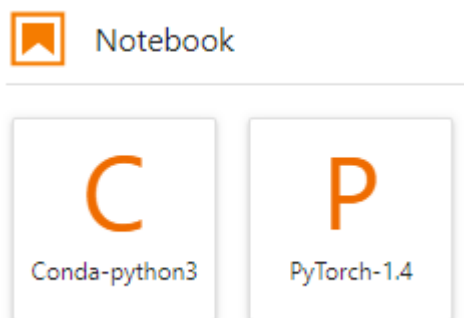
```
pip install Shapely
```

Creating an IPYNB File in JupyterLab

On the JupyterLab homepage, click a proper AI engine in the **Notebook** area to create an IPYNB file.

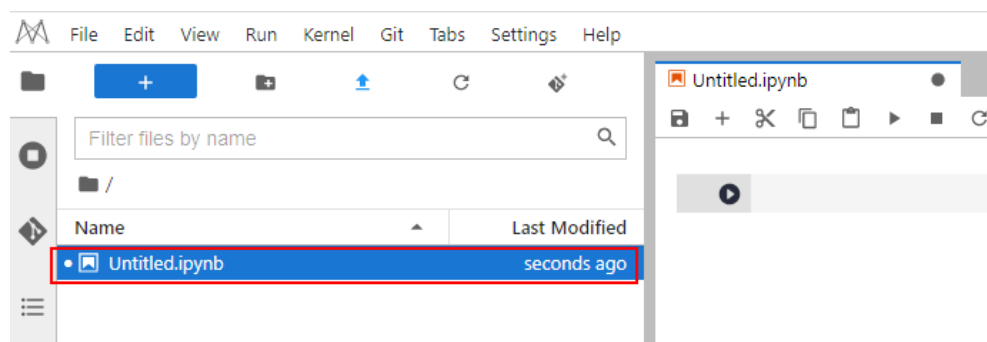
The AI engines supported by each notebook instance vary depending on the runtime environment. The following figure is only an example. Select an AI engine based on site requirements.

Figure 5-35 Selecting an AI engine and creating IPYNB file



The created IPYNB file is displayed in the navigation pane on the left.

Figure 5-36 Created IPYNB file



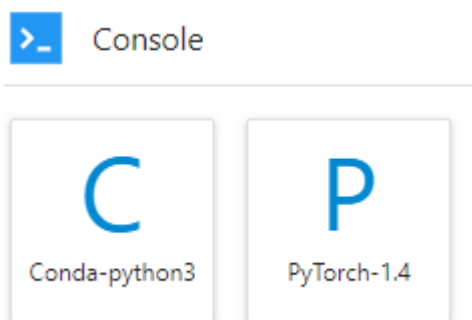
Creating a Notebook File and Accessing the Console

A console is a Python terminal, which is similar to the native IDE of Python, displaying the output after a statement is entered.

On the JupyterLab homepage, click a proper AI engine in the **Console** area to create a notebook file.

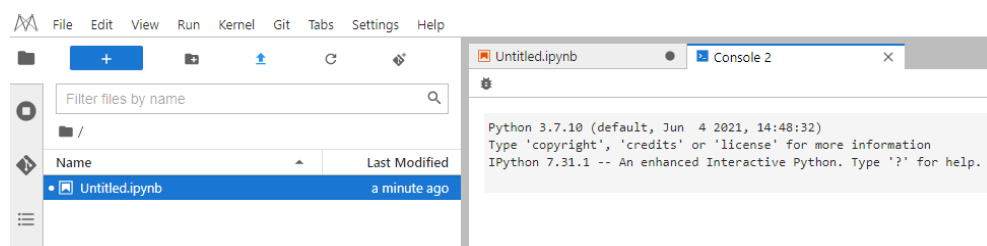
The AI engines supported by each notebook instance vary depending on the runtime environment. The following figure is only an example. Select an AI engine based on site requirements.

Figure 5-37 Selecting an AI engine and creating a console



After the file is created, the console page is displayed.

Figure 5-38 Creating a notebook file (console)

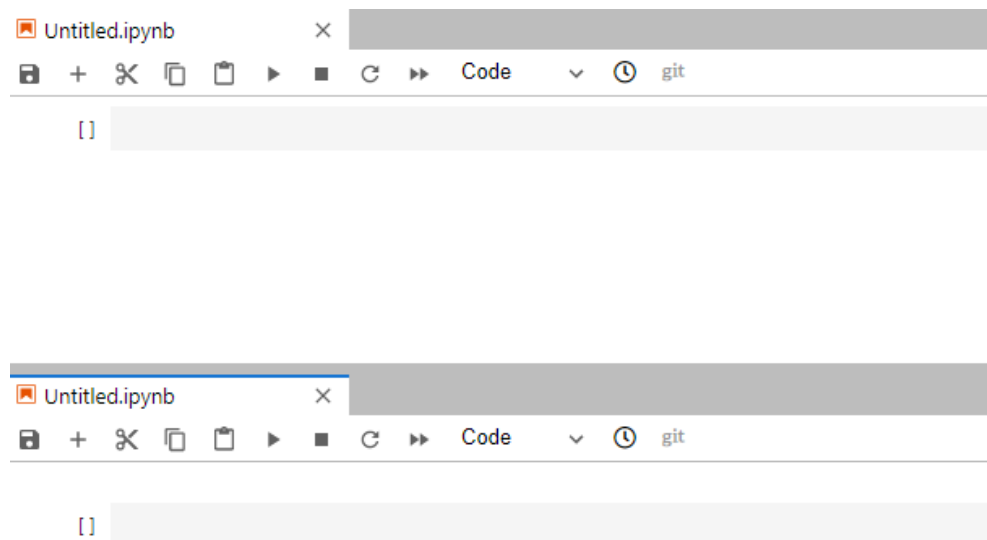


Editing a File in JupyterLab

JupyterLab allows you to open multiple notebook instances or files (such as HTML, TXT, and Markdown files) in one window and displays them on different tab pages.

In JupyterLab, you can customize the display of multiple files. In the file display area on the right, you can drag a file to adjust its position. Multiple files can be concurrently displayed.

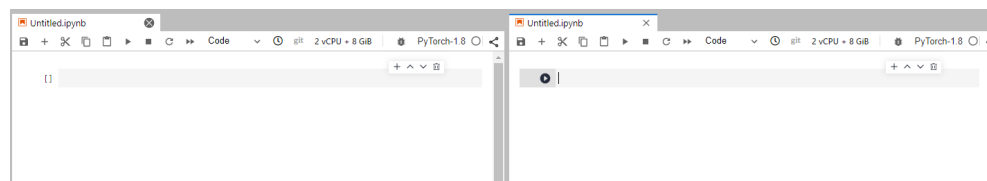
Figure 5-39 Customized display of multiple files



When writing code in a notebook instance, you can create multiple views of a file to synchronously edit the file and view execution results in real time.

To open multiple views, open an IPYNB file and choose **File > New View for Notebook**.

Figure 5-40 Multiple views of a file



Before coding in the code area of an IPYNB file in JupyterLab, add an exclamation mark (!) before the code.

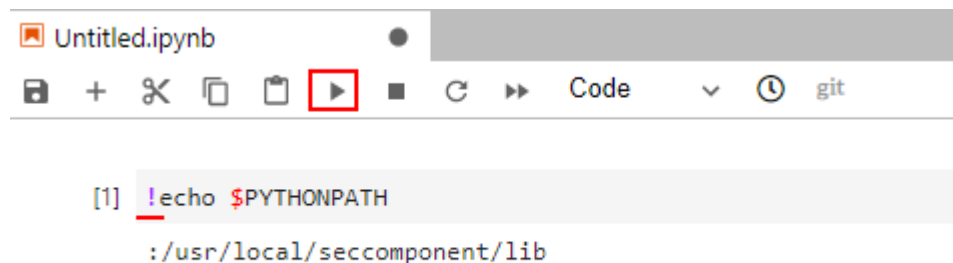
For example, install an external library Shapely.

```
!pip install Shapely
```


For example, obtain PythonPath.

```
!echo $PYTHONPATH
```

Figure 5-41 Running code



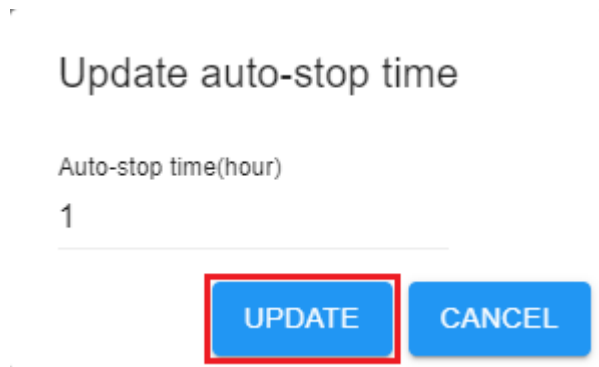
Renewing or Automatically Stopping a Notebook Instance

If you enable auto stop when you created or started a notebook instance, the remaining duration for stopping the instance is displayed in the upper right corner of JupyterLab. You can click the time for renewal.

Figure 5-42 Remaining duration



Figure 5-43 Renewing an instance



Common JupyterLab Buttons and Plug-ins

Figure 5-44 Common JupyterLab buttons and plug-ins

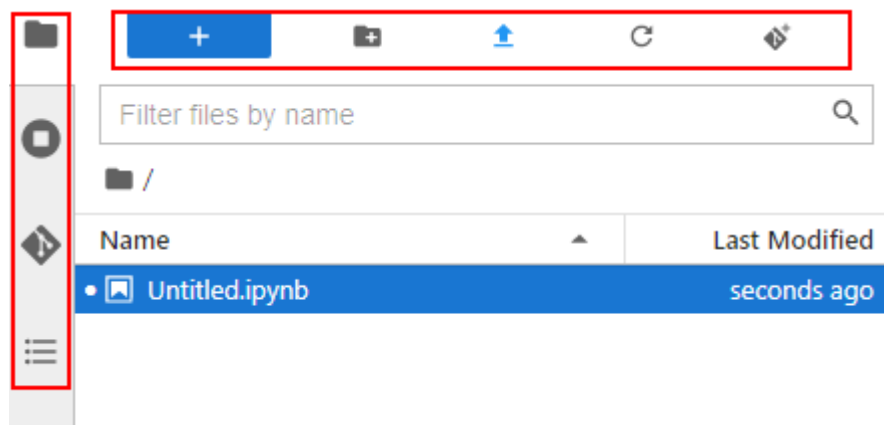


Table 5-18 JupyterLab buttons

Button	Description
	Quickly open notebook instances and terminals. Open the Launcher page, on which you can quickly create notebook instances, consoles, or other files.
	Create a folder.
	Upload files.
	Refresh the file directory.
	Git plug-in, which can be used to access the GitHub code library associated with the notebook instance.

Table 5-19 JupyterLab plug-ins

Plug-in	Description
	List files. Click this button to show all files in the notebook instance.
	Display the terminals and kernels that are running in the current instance.
	Git plug-in, which can be used to quickly access the GitHub code library.
	Property inspector.

Plug-in	Description
	Show the document organization.

Figure 5-45 Buttons in the navigation bar



Table 5-20 Buttons in the navigation bar

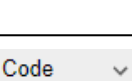
Button	Description
File	Actions related to files and directories, such as creating, closing, or saving notebooks.
Edit	Actions related to editing documents and other activities in the IPYNB file, such as undoing, redoing, or cutting cells.
View	Actions that alter the appearance of JupyterLab, such as showing the bar or expanding code.
Run	Actions for running code in different activities such as notebooks and code consoles.
Kernel	Actions for managing kernels, such as interrupting, restarting, or shutting down a kernel.
Git	Actions on the Git plug-in, which can be used to quickly access the GitHub code library.
Tabs	A list of the open documents and activities in the dock panel.
Settings	Common settings and an advanced settings editor.
Help	A list of JupyterLab and kernel help links.

Figure 5-46 Buttons in the menu bar of an IPYNB file



Table 5-21 Buttons in the menu bar of an IPYNB file

Button	Description
	Save a file.

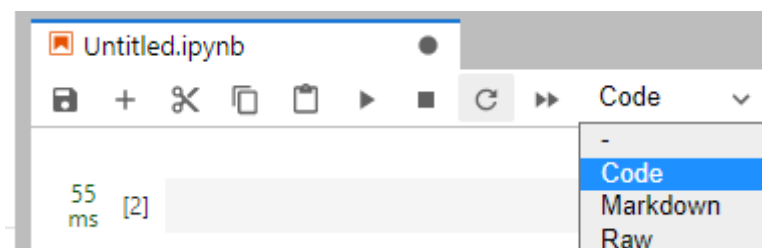
Button	Description
	Add a new cell.
	Cut the selected cell.
	Copy the selected cell.
	Paste the selected cell.
	Execute the selected cell.
	Terminate a kernel.
	Restart a kernel.
	Restart a kernel and run all code of the current notebook again.
	There are four options in the drop-down list: Code (Python code), Markdown (Markdown code, typically used for comments), Raw (a conversion tool), and - (not modified)
	View historical code versions.
	Git plug-in. The gray button indicates that the plug-in is unavailable in the current region.
	Instance flavor.
	Kernel for you to select.
	Code running status.  indicates the code is being executed.

Using Code-based Plug-ins

The code parametrization plug-in simplifies notebook cases. You can quickly adjust parameters and train models based on notebook cases without complex code. This plug-in can be used to customize notebook cases for competitions and learning.

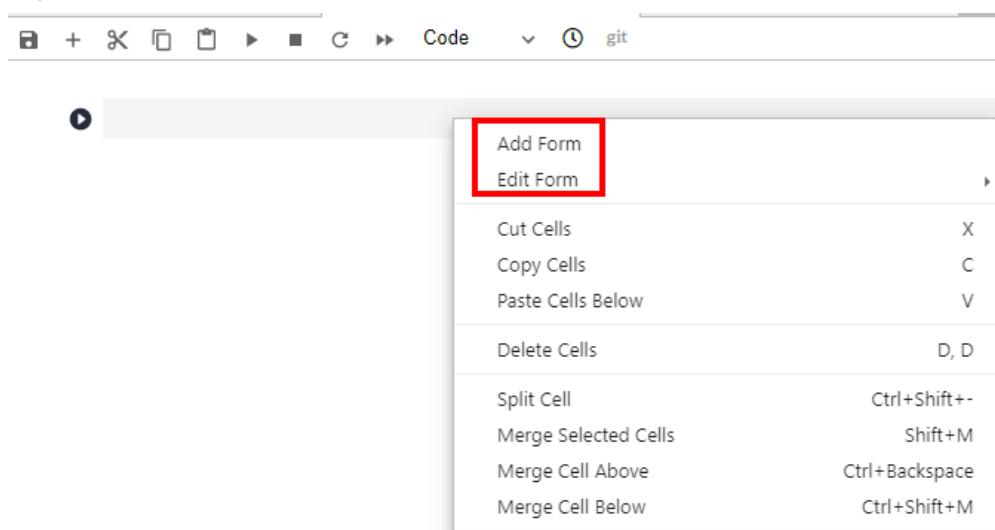
- The **Add Form** and **Edit Form** buttons are available only to the shortcut menu of code cells, as shown below.

Figure 5-47 Viewing a code cell



- After opening new code, add a form before editing it.

Figure 5-48 Shortcut menu of code cells



- If you click **Add Form**, a code cell will be split into the code and form edit area. Click **Edit** on the right of the form to change the default title.

Figure 5-49 Two edit areas



- If you click **Edit Form**, four sub-options will be displayed: **Add new form field**, **Hide code**, **Hide form**, and **Show All**.

Table 5-22 Edit Form sub-options

Edit-Form Sub-option	Description
Add new form field	- The form field types include dropdown, input, and slider, as shown in Figure 5-50. Each time a field is added, the corresponding variable is added to the code and form areas. If a value in the form area is changed, the corresponding variable in the code area is also changed. NOTE When creating a dropdown form, click ADD Item and add at least two items, as shown in Figure 5-51. - If the form field type is set to dropdown, the supported variable types are raw and string. - If the form field type is set to input, the supported variable types are boolean, date, integer, number, raw, and string. - If the form field type is set to slider, the minimum value, maximum value, and step can be set.
Hide code	Hide the code.
Hide form	Hide the forms.
Show all	Display both code and forms.

Figure 5-50 Form field types

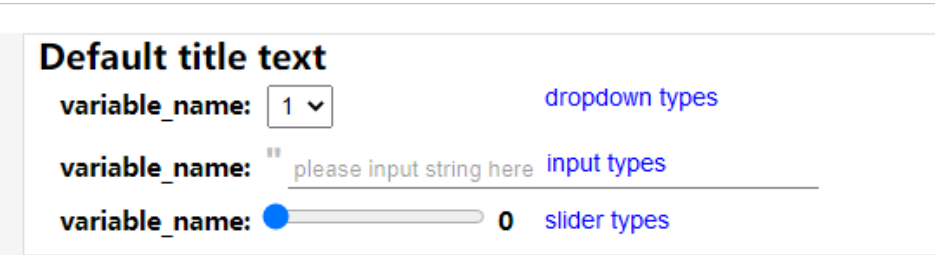


Figure 5-51 Creating a dropdown form

Add new form field

Form field type: dropdown

Variable type: string

Variable name: variable_name

Buttons: Cancel, Save

Red box highlights:

- Add Item
- option1 input a value
- option2 input a value

Figure 5-52 Deleting a form

Default title text

variable_name1: bb

variable_name2: [1, 2, 3]

variable_name3: {'1': 'a', 'b':2}

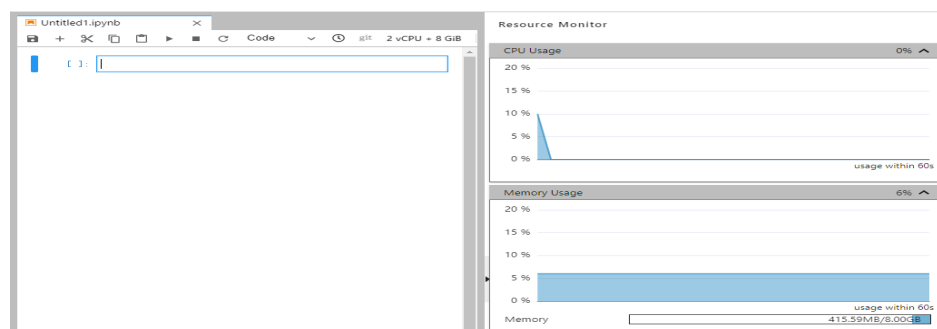
variable_name4: (1, 2, 3)

Red box highlights the delete icon in the toolbar.

Monitoring Resources

To obtain resource usage, select **Resource Monitor** in the right pane. The CPU usage and memory usage can be viewed.

Figure 5-53 Resource usage



5.5.3 Using Git to Clone the Code Repository in JupyterLab

In JupyterLab, you can use the Git plug-in to clone the GitHub open-source code repository, quickly view and edit data, and submit the modified data.

Prerequisites

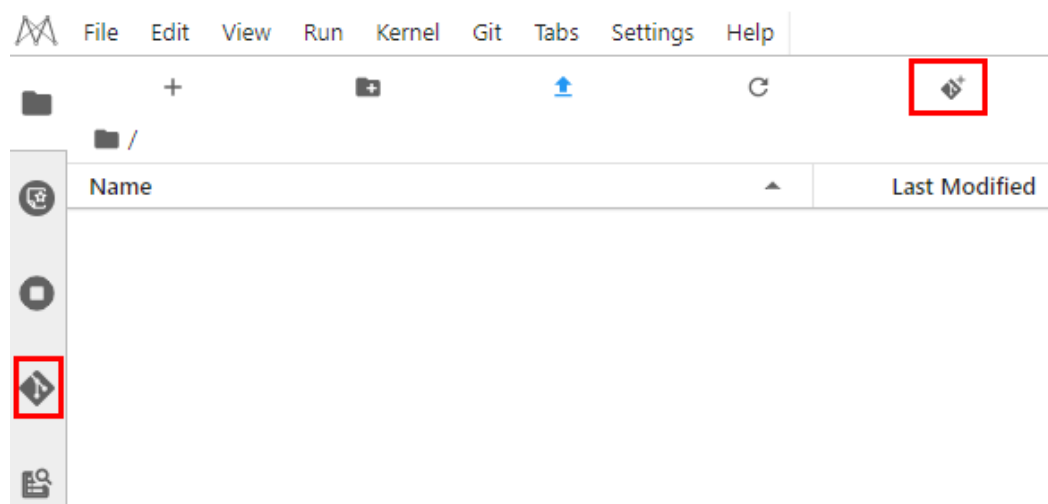
The notebook instance is running.

Starting the Git Plug-in of JupyterLab

In the notebook instance list, locate the target instance and click **Open** in the **Operation** column to go to the JupyterLab page.

Figure 5-54 shows the Git plug-in of JupyterLab.

Figure 5-54 Git plug-in



Cloning a GitHub Open-Source Code Repository

Access a GitHub open-source code repository at <https://github.com/jupyterlab/extension-examples>. Click , enter the repository address, and click **OK** to start cloning. After the cloning is complete, the code library folder is displayed in the navigation pane of JupyterLab.

Cloning a GitHub Private Code Repository

1. When you clone a GitHub private code repository, a dialog box will be displayed, asking you to enter your personal credentials. In this case, enter the personal access token in GitHub.

Git credentials required

Enter credentials for remote repository

2. To obtain a personal access token, perform the following operations:
 - a. Log in to [GitHub](#) and open the configuration page.
 - b. Click **Developer settings**.
 - c. Choose **Personal access tokens > Generate new token**.
 - d. Verify the login account.
 - e. Describe the token, select permissions to access the private repository, and click **Generate token** to generate a token.
 - f. Copy the generated token to CloudBuild.

NOTICE

- Save the token securely once it is generated. It will be unavailable after you refresh the page. If it is not obtained, generate a new token.
- Enter a valid token description so that it can be easily identified. If the token is deleted by mistake, the building will fail.
- Delete the token when it is no longer used to prevent information leakage.

Figure 5-55 Cloning a GitHub private code repository (only authorization using a personal access token is supported)

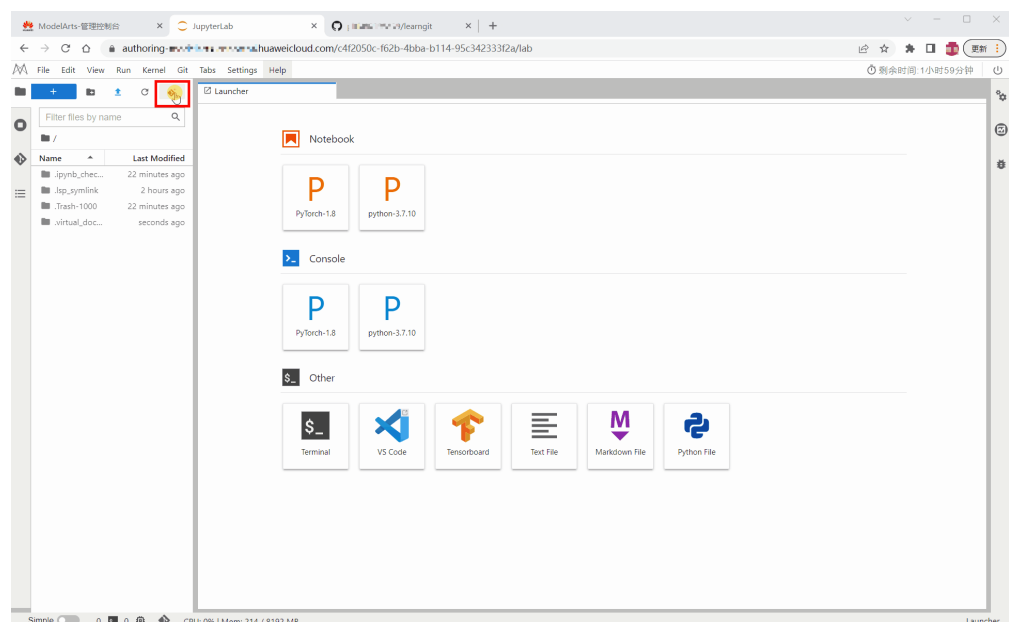
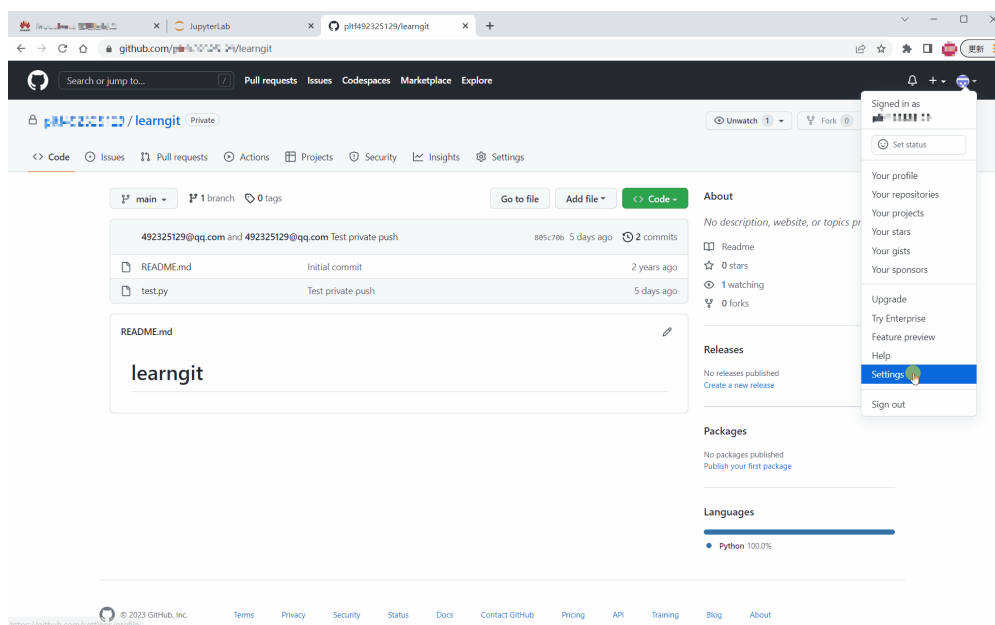


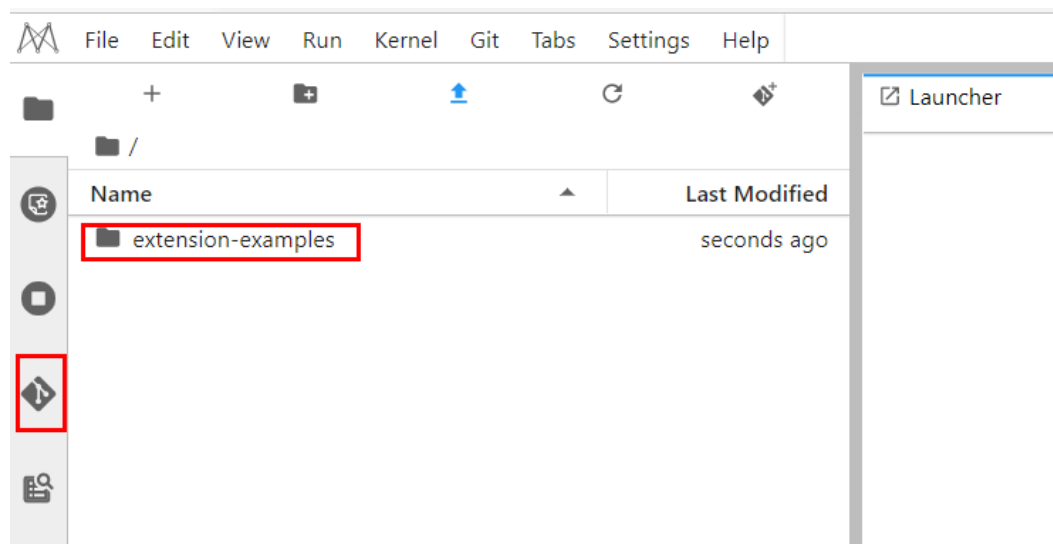
Figure 5-56 Obtaining a personal access token



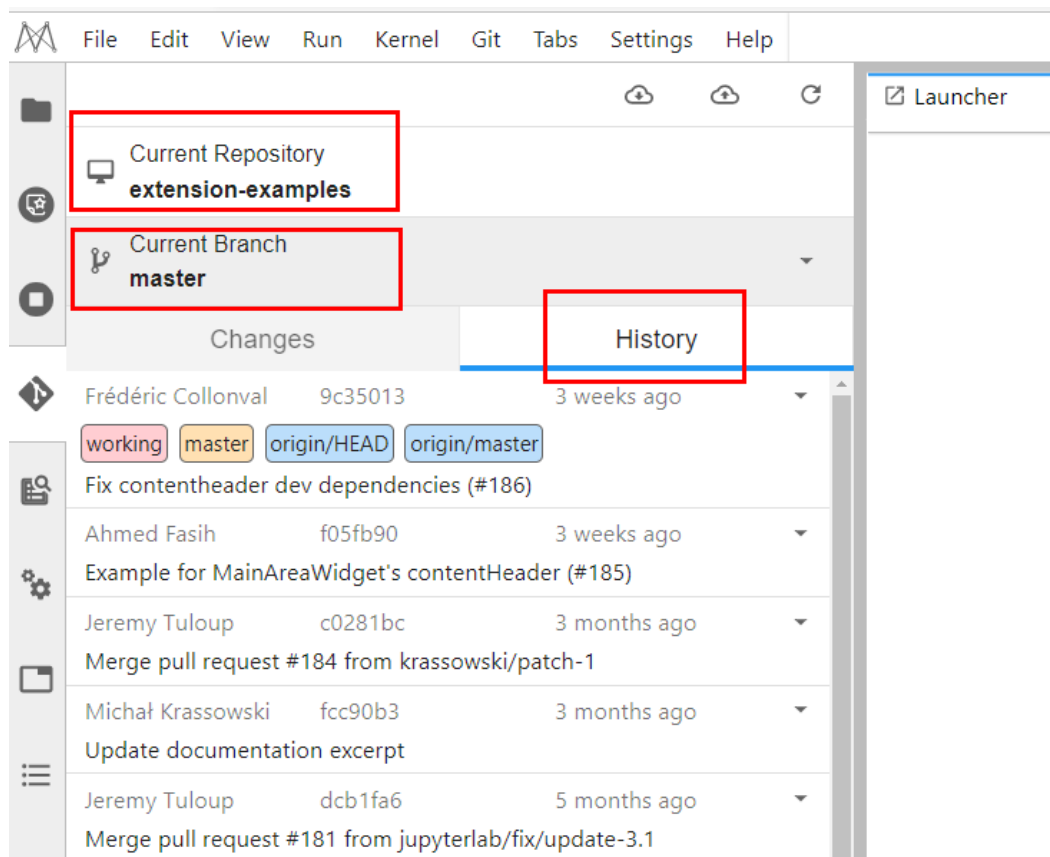
Viewing a Code Repository

In the list under **Name**, double-click the folder you want to use and click the Git plug-in icon on the left to access the code repository corresponding to the folder.

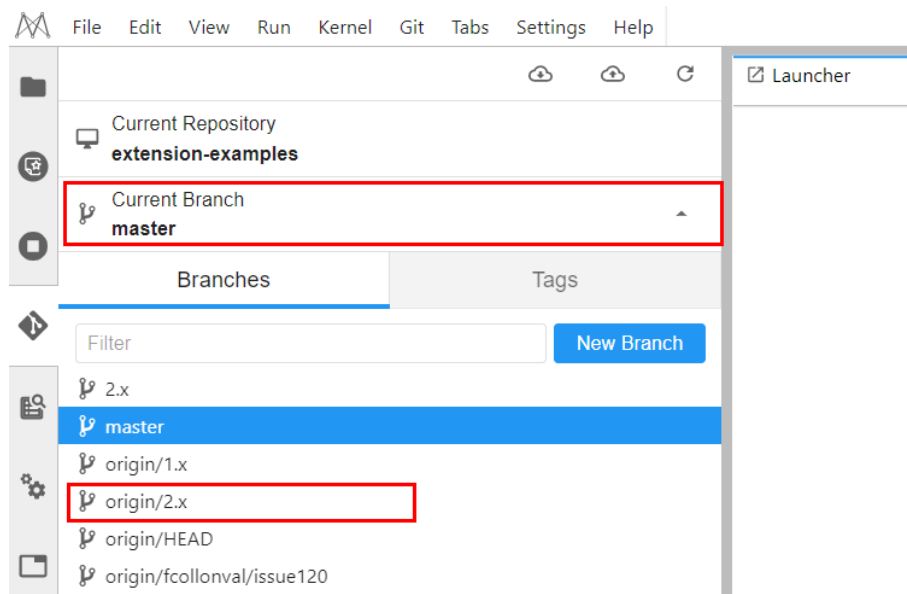
Figure 5-57 Opening the folder and starting the Git plug-in



You can view the information current code repository, such as the repository name, branch, and historical submission records.

Figure 5-58 Viewing a code repository**NOTE**

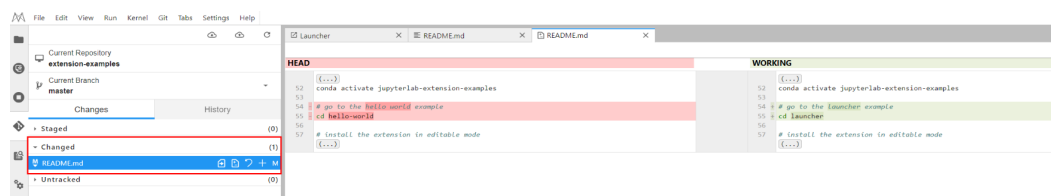
By default, the Git plug-in clones the master branch. To switch another branch, click **Current Branch** to expand all branches and click the target branch name.



Viewing Modifications

If a file in the code repository has been modified, you can view the modified file under **Changed** on the **Changes** tab page. Click **Diff this file** on the right of the file name to view the modifications.

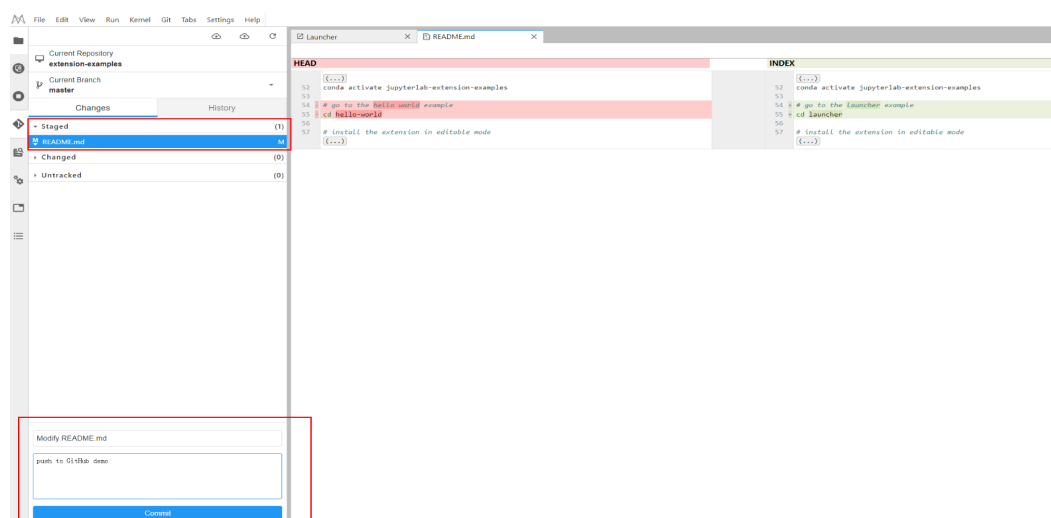
Figure 5-59 Viewing modifications



Committing Modifications

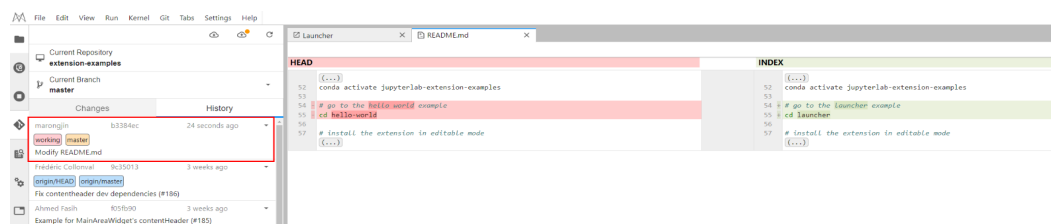
After confirming that the modifications are correct, click **Stage this change** on the right of the file name, which is equivalent to running the **git add** command. The file enters the **Staged** state. Enter the message to be committed in the lower left corner and click **Commit** that is equivalent to running the **git commit** command.

Figure 5-60 Committing modifications



On the **History** tab page, view the committing status.

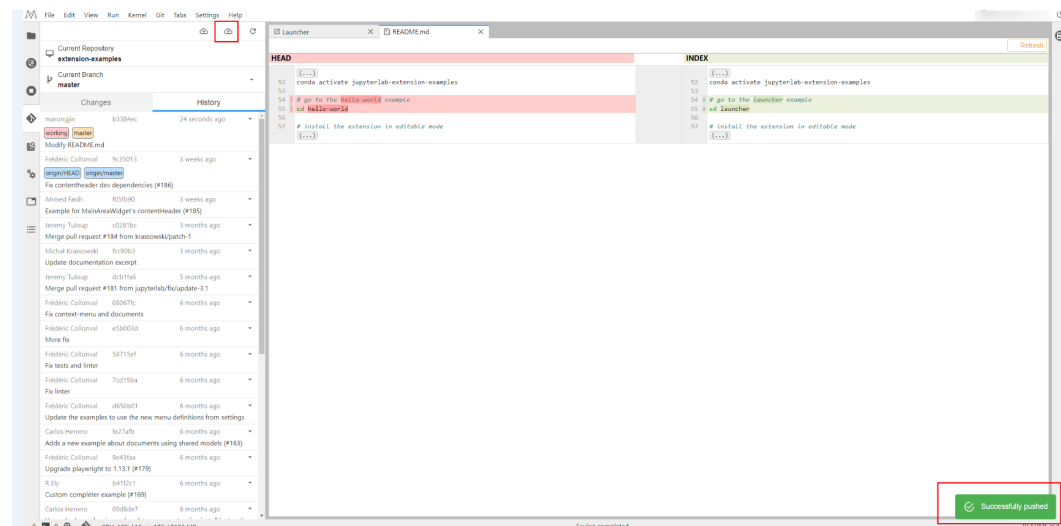
Figure 5-61 Checking whether the committing is successful



Click the **push** icon, which is equivalent to running the **git push** command, to push the code to the GitHub repository. After the pushing is successful, the

message "Successfully completed" is displayed. If the token used for OAuth authentication has expired, a dialog box is displayed asking you to enter the user token or account information. Enter the information as prompted. This section describes the authorization using a personal access token. If you use a password for authorization but the password becomes unavailable, perform the operations described in [What Do I Do If the Git Plug-in Password Is Invalid?](#)

Figure 5-62 Pushing code to the GitHub repository



After the preceding operations are complete, on the **History** tab page of the JupyterLab Git plug-in page, you can see that **origin/HEAD** and **origin/master** point to the latest push. In addition, you can find the corresponding information in the committing records of the GitHub repository.

5.5.4 Creating a Scheduled Job in JupyterLab

You can create a scheduled job in a ModelArts notebook instance. This section describes how to create a scheduled job and run a notebook file with one click to improve efficiency.

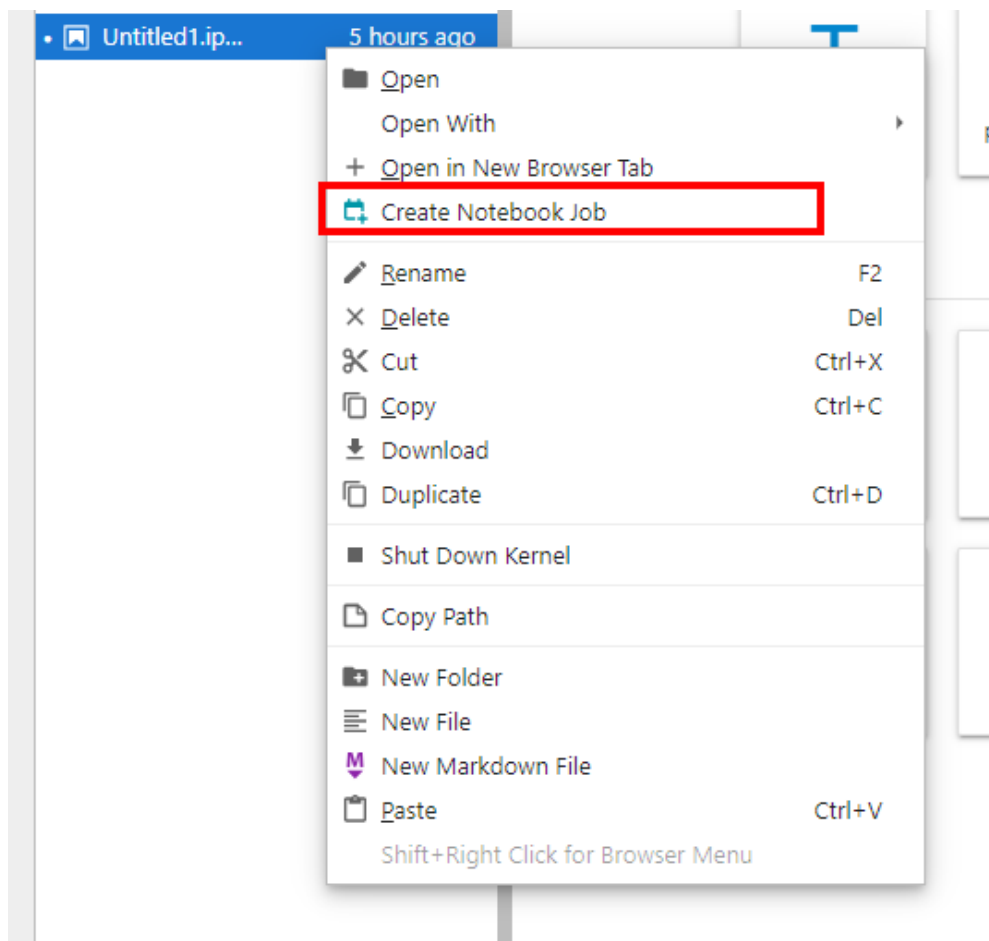
Highlights

- **One-click running:** You can run a notebook file with one click.
- **Scheduled task:** You can set the time and frequency for executing code blocks. The time can be set by second, minute, hour, day, week, or month.
- **Parameter-based execution:** You can transfer parameters to a notebook job during its runtime, so that the notebook job can adjust its behavior as required.
- **Task management GUI:** A user-friendly GUI is provided for viewing, adding, and deleting scheduled tasks.
- **Task execution record:** The status and output of each job are recorded for future query and debugging.

Procedure

1. Open ModelArts Notebook.
2. Choose a notebook file (IPYNV file) to create a scheduled job.

Figure 5-63 Opening notebook jobs



3. On the **Create Job** page, configure the parameters and click **Create**.

Figure 5-64 Parameters for creating a scheduled job

The screenshot shows the 'Create Job' interface in the ModelArts Notebook Jobs section. The interface is divided into several sections:

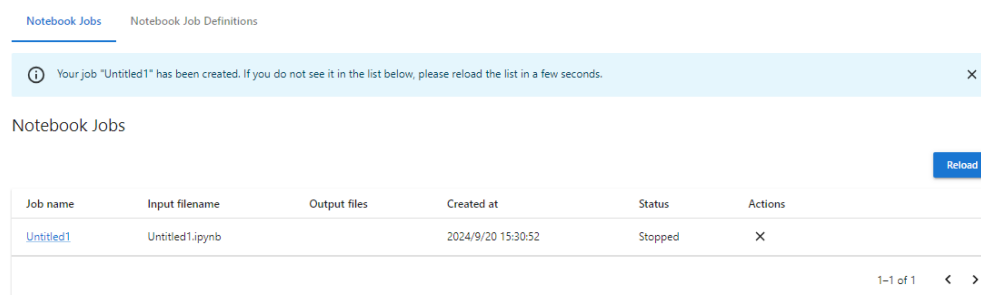
- Job name:** A text input field containing 'Untitled'.
- Input file:** A text input field containing '/Untitled.ipynb'.
- Environment:** A dropdown menu showing 'anaconda3'.
- Output formats:** Two checkboxes, 'Notebook' and 'HTML', both of which are checked.
- Parameters:** A section with a plus sign (+) to add parameters.
- Additional options:** A dropdown menu.
- Schedule:** Two radio buttons, 'Run now' and 'Run on a schedule'. 'Run on a schedule' is selected.
- Interval:** A dropdown menu showing 'Custom schedule'.
- cron expression:** A text input field containing '00 ** 1'.
- Time zone:** A dropdown menu showing 'Asia/Shanghai'.
- Buttons:** 'Cancel' and 'Create' buttons at the bottom right.

- **Job name:** Enter the scheduled job name.
- **Environment:** Choose a Python environment where the notebook job is to be run.
- **Output formats:** Select the output file type of the execution result
- **Parameters:** Click the plus sign (+) to set the Python variables for running the job.

- **Schedule:** Choose **Run now** or **Run on a schedule**. The cron expression is supported.

NOTE

- The cron expression must be in the format supported by Linux OS. Otherwise, an error will be reported. If the expression contains a question mark (?), replace it with an asterisk (*).
 - After a scheduled job is configured, modify the file name and content. The created jobs are not affected.
4. Run the job. You can view the job running records in the **Notebook Jobs** tab. Click **Reload** in the upper right corner to get the latest records.

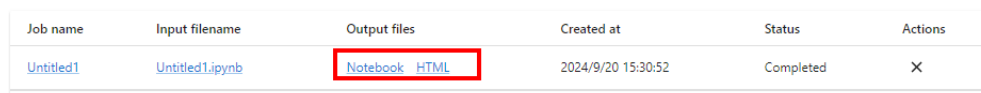
Figure 5-65 Viewing the running records of scheduled jobs

Notebook Jobs

Job name	Input filename	Output files	Created at	Status	Actions
Untitled1	Untitled1.ipynb		2024/9/20 15:30:52	Stopped	×

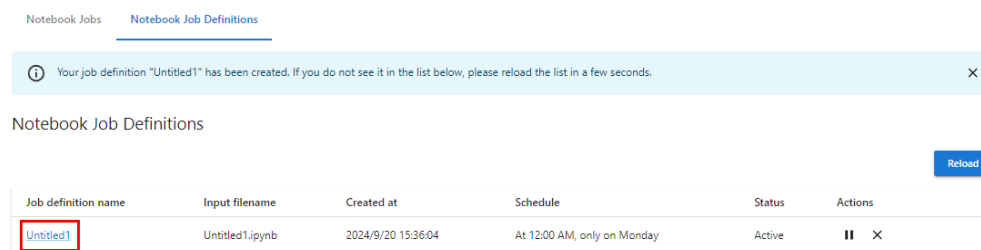
1-1 of 1 < >

5. After the job is executed, you can download the output files. Click the file name to view the execution result.

Figure 5-66 Viewing the execution result of a scheduled job

Job name	Input filename	Output files	Created at	Status	Actions
Untitled1	Untitled1.ipynb	Notebook HTML	2024/9/20 15:30:52	Completed	×

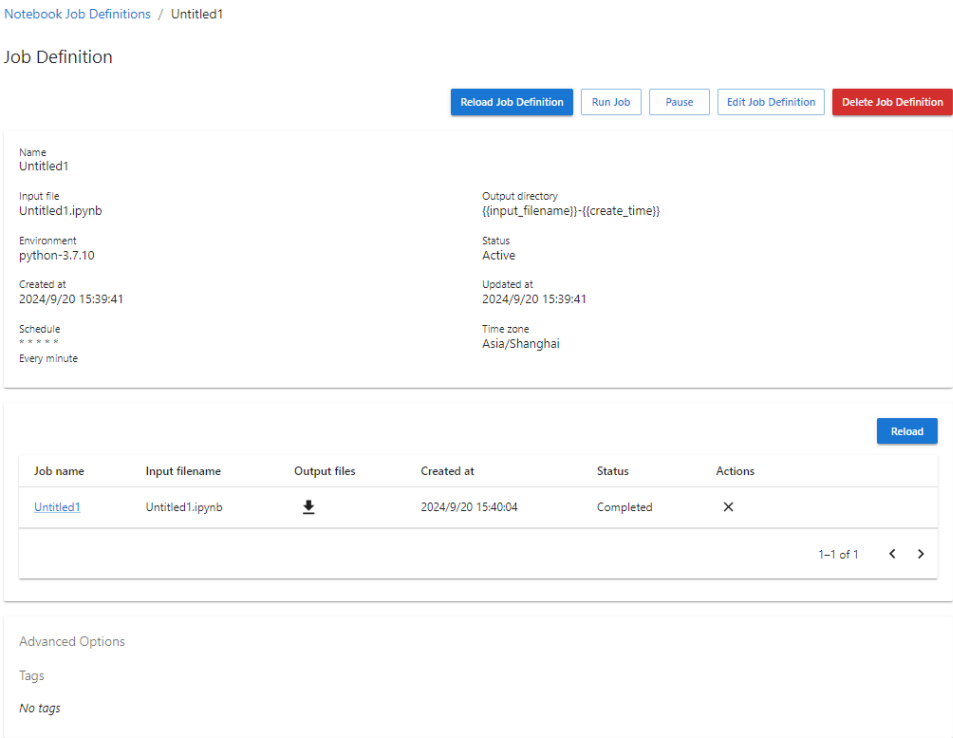
6. In the **Notebook Job Definitions** tab, view all jobs. Click a job name to access the **Job Definition** page. You can start, stop, or delete a scheduled job. You can also click **Edit Job Definition** to update the scheduling information or view the running history of a scheduled job.

Figure 5-67 Notebook Job Definitions tab

Notebook Job Definitions

Job definition name	Input filename	Created at	Schedule	Status	Actions
Untitled1	Untitled1.ipynb	2024/9/20 15:36:04	At 12:00 AM, only on Monday	Active	⏸ ×

Figure 5-68 Configuring a scheduled job



5.5.5 Uploading Files to JupyterLab

5.5.5.1 Uploading Files from a Local Path to JupyterLab

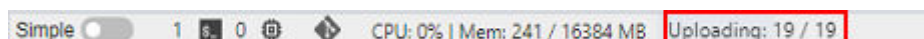
JupyterLab provides multiple methods for uploading files.

Methods for Uploading a File

- For a file not exceeding 100 MB, directly upload it to the target notebook instance. Detailed information, such as the file size, upload progress, and upload speed are displayed.
- For a file that exceeds 100 MB but does not exceed 50 GB, upload the file to OBS (an object bucket or a parallel file system), and then download the file from OBS to a notebook instance. After the download is complete, the file is deleted from OBS.
- For a file that exceeds 50 GB, upload it by calling ModelArts SDK or MoXing.
- For a file that shares the same name with an existing file in the current directory of a notebook instance, overwrite the existing file or cancel the upload.
- A maximum of 10 files can be uploaded at a time. The other files are in awaiting upload state. No folders can be uploaded. If a folder is required, compress it into a package, upload the package to notebook, and decompress the package in Terminal.
unzip xxx.zip # Directly decompress the package in the path where the package is stored.

For more details, search for the decompression command in mainstream search engines.

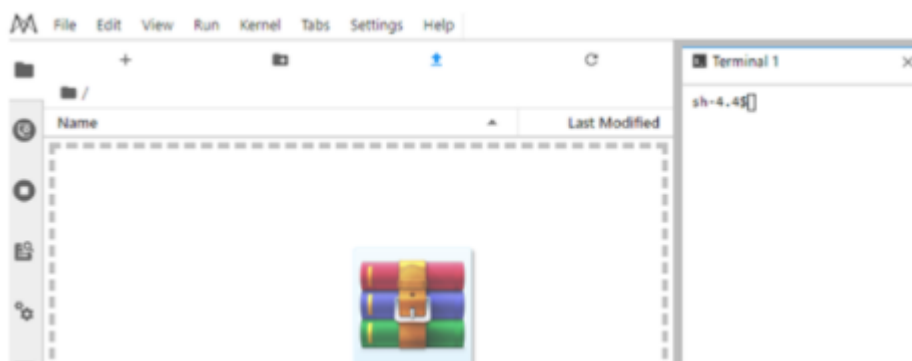
- When multiple files are uploaded in a batch, the total number of files to be uploaded and the number of files that have been uploaded are displayed at the bottom of the JupyterLab window.



Uploading File

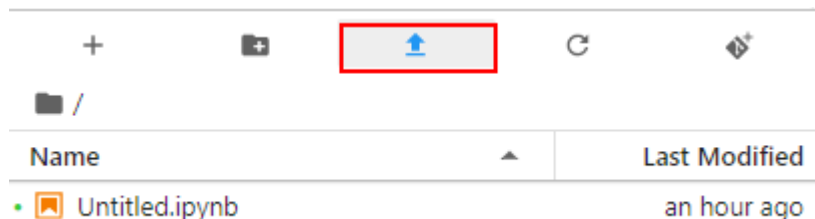
Method 1: Use JupyterLab to open a running notebook environment.

Figure 5-69 Dragging the file to JupyterLab



Method 2: Click  in the navigation bar on the top of the window. In the displayed dialog box, drag or select a local file and upload it.

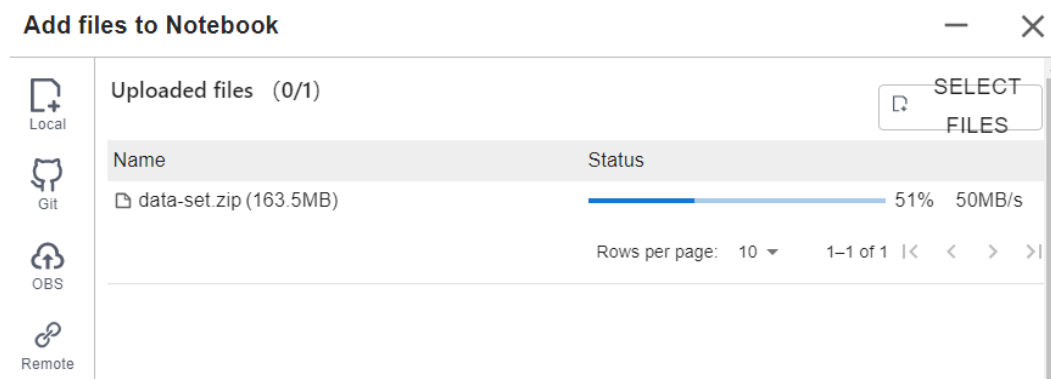
Figure 5-70 Clicking the upload button



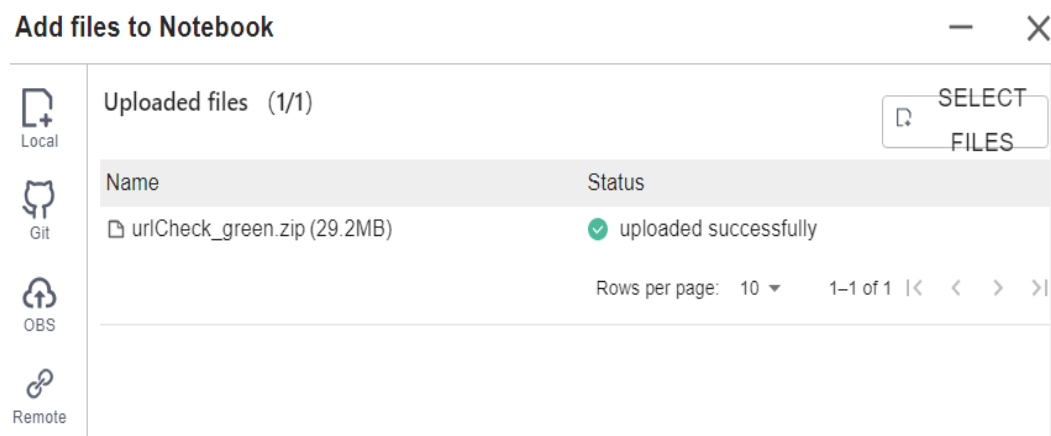
Name	Last Modified
•  Untitled.ipynb	an hour ago

Uploading a Local File Smaller Than 100 MB to JupyterLab

For a file not exceeding 100 MB, directly upload it to the target notebook instance. Detailed information, such as the file size, upload progress, and upload speed are displayed.

Figure 5-71 Uploading a file less than 100 MB

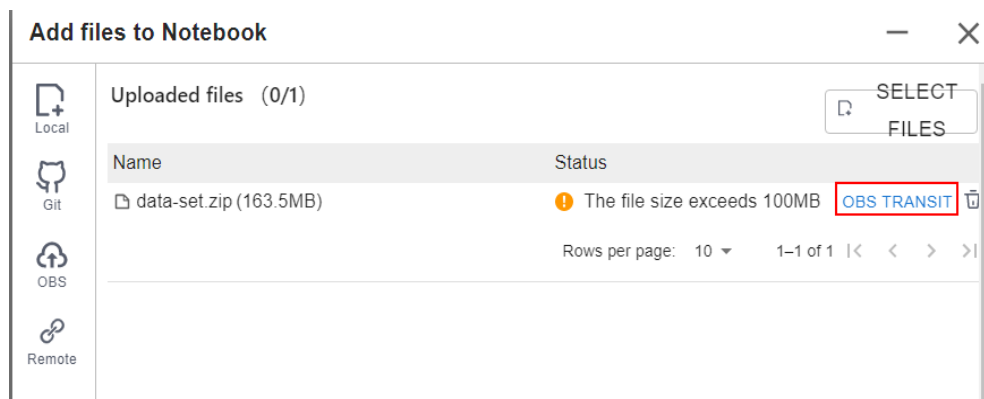
A message is displayed after the file is uploaded.

Figure 5-72 Uploaded

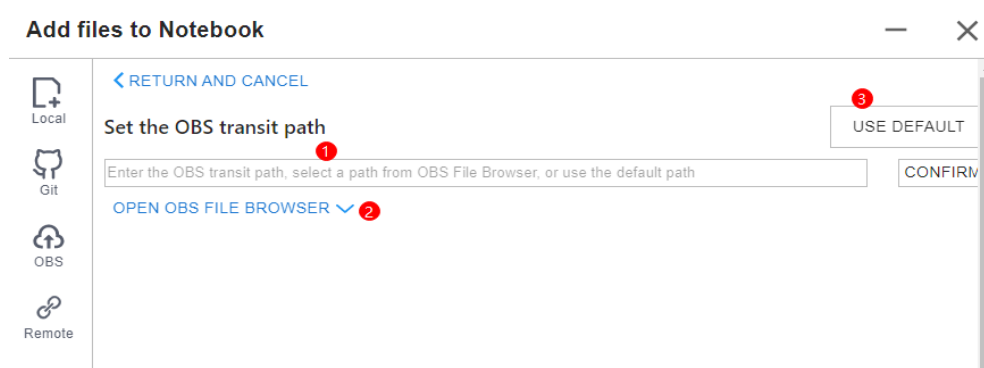
Uploading a Local File that Is 100 MB to 50 GB to JupyterLab

For a file that exceeds 100 MB but does not exceed 50 GB, upload the file to OBS (an object bucket or a parallel file system), and then download the file from OBS to the target notebook instance. After the download is complete, the file is automatically deleted from OBS.

For example, in the scenario shown in the following figure, upload the file through OBS.

Figure 5-73 Uploading a large file through OBS

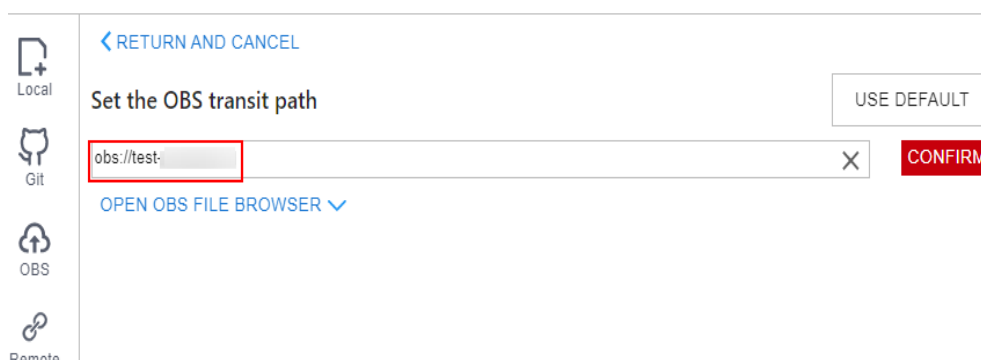
To upload a large file through OBS, set an OBS path.

Figure 5-74 Uploading a file through OBS

NOTE

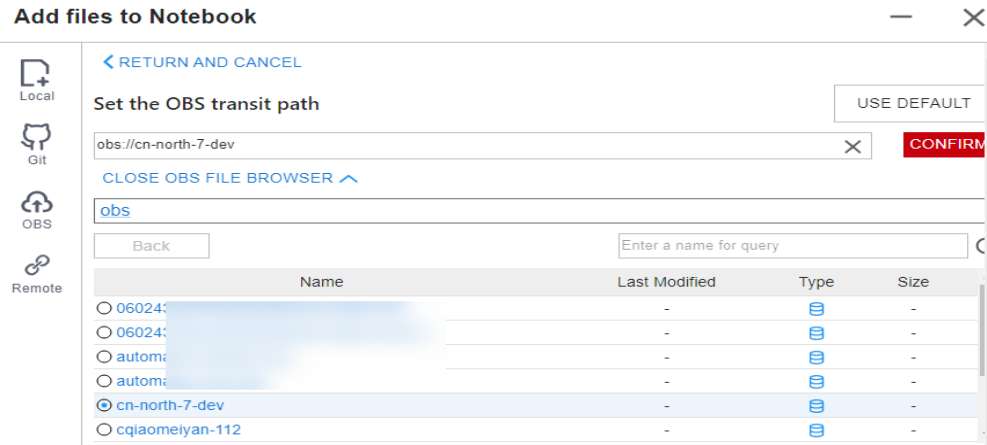
Set an OBS path for uploading local files to JupyterLab. After the setting, this path is used by default in follow-up operations. To change the path, click in the lower left corner of the file upload window, as shown in [Figure 5-78](#).

- Method 1: Enter a valid OBS path in the text box and click **OK**.

Figure 5-75 Configuring an OBS path

- Method 2: Select an OBS path in **OBS File Browser** and click **OK**.

Figure 5-76 OBS File Browser



- Method 3: Use the default path.

Figure 5-77 Using the default path to upload a file

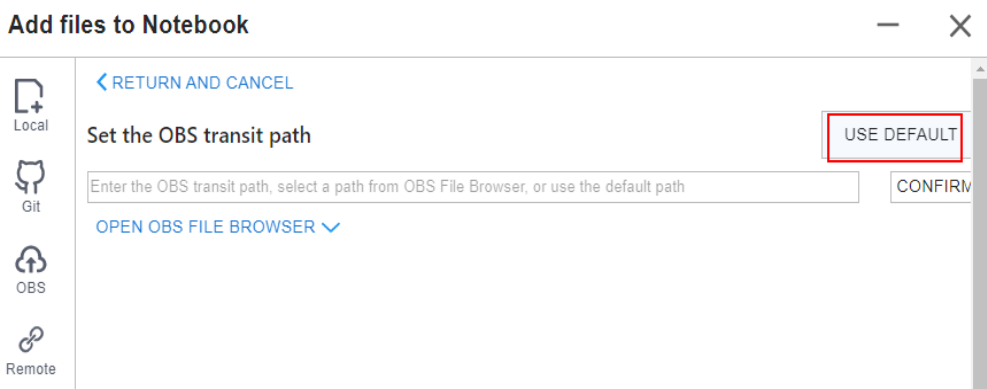
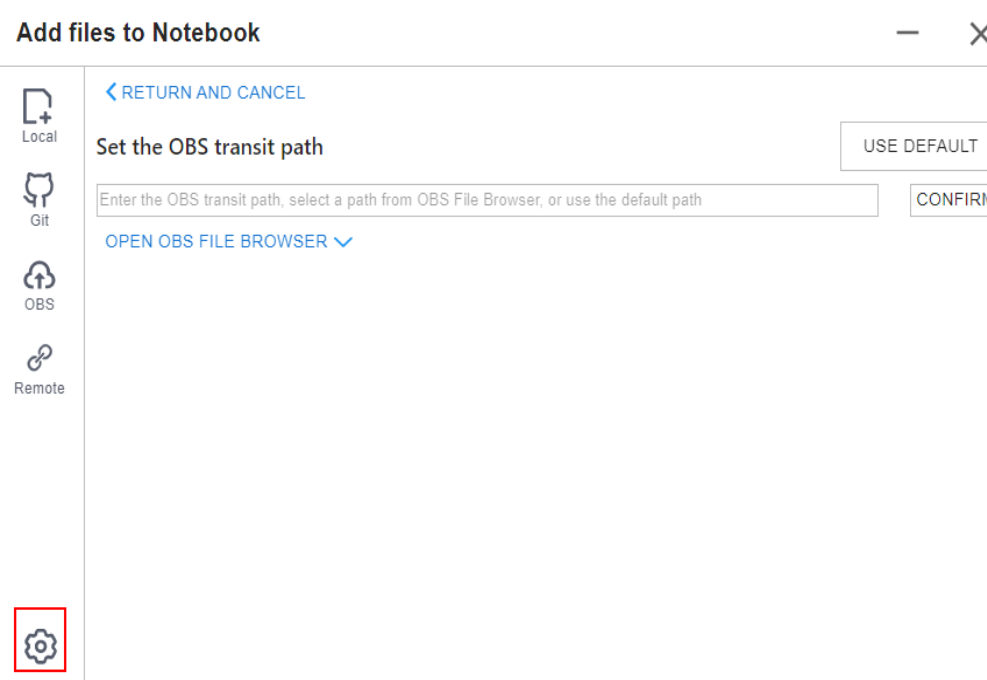
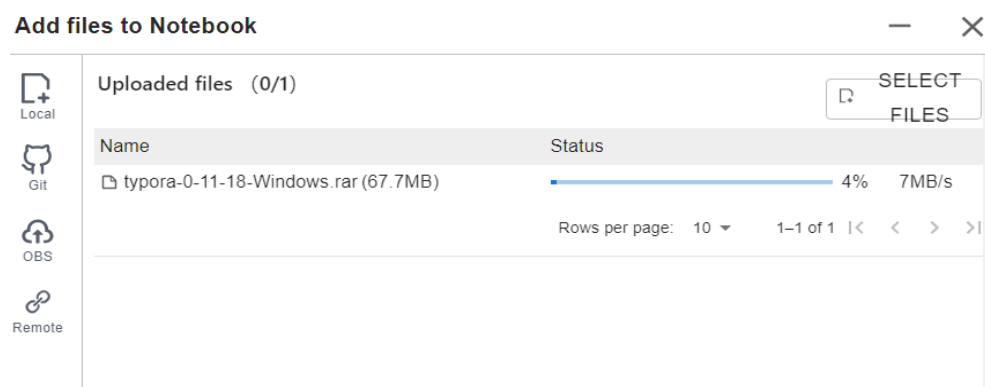


Figure 5-78 Setting an OBS path to upload a local file



After the OBS path is set, upload a file.

Figure 5-79 Uploading a file



Decompressing a package

After a large file is uploaded to Notebook JupyterLab as a compressed package, you can decompress the package in Terminal.

```
unzip xxx.zip # Directly decompress the package in the path where the package is stored.
```

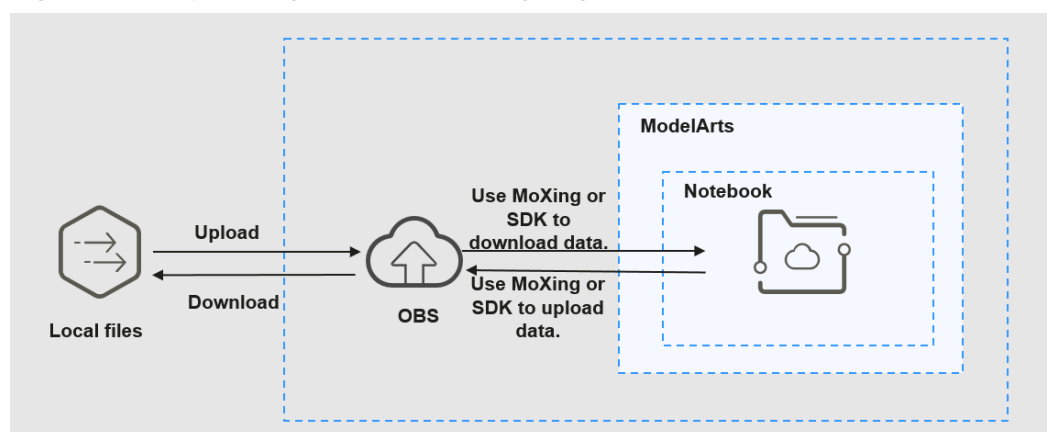
For more details, search for the decompression command in mainstream search engines.

Uploading a Local File Larger Than 50 GB to JupyterLab

A file exceeding 50 GB cannot be directly uploaded to JupyterLab.

To upload files exceeding 50 GB, upload them to OBS. Then, call the ModelArts MoXing or SDK API in the target notebook instance to read and write the files in OBS.

Figure 5-80 Uploading and downloading large files in a notebook instance



The procedure is as follows:

1. Upload the file from a local path to OBS. For details, see [Uploading an Object](#).

2. Download the file from OBS to the notebook instance by calling the ModelArts SDK or MoXing API.

- Method 1: Call the ModelArts SDK to download a file from OBS.

Example code:

```
from modelarts.session import Session
session = Session()
session.obs.copy("obs://bucket-name/obs_file.txt", "/home/ma-user/work/")
```

- Method 2: Call the ModelArts MoXing API for reading an OBS file.

import moxing as mox

```
# Download the OBS folder sub_dir_0 from OBS to a notebook instance.
mox.file.copy_parallel('obs://bucket_name/sub_dir_0', '/home/ma-user/work/sub_dir_0')
# Download the OBS file obs_file.txt from OBS to a notebook instance.
mox.file.copy('obs://bucket_name/obs_file.txt', '/home/ma-user/work/obs_file.txt')
```

If a .zip file is downloaded, run the following command on the terminal to decompress the package:

```
unzip xxx.zip # Directly decompress the package in the path where the package is stored.
```

After the code is executed, open the terminal shown in [Figure 5-81](#) and run the **ls /home/ma-user/work** command to view the file downloaded to the notebook instance. Alternatively, go to JupyterLab. In the navigation pane on the left, view the downloaded file. If the file is not displayed, refresh the page, as shown in [Figure 5-82](#).

Figure 5-81 Terminal

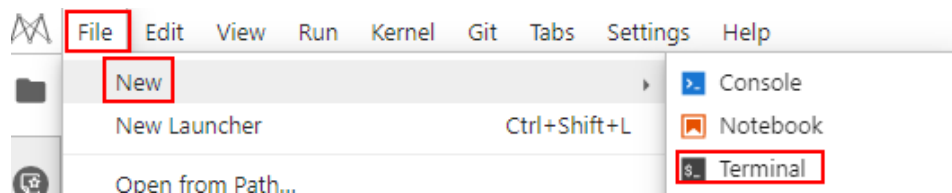
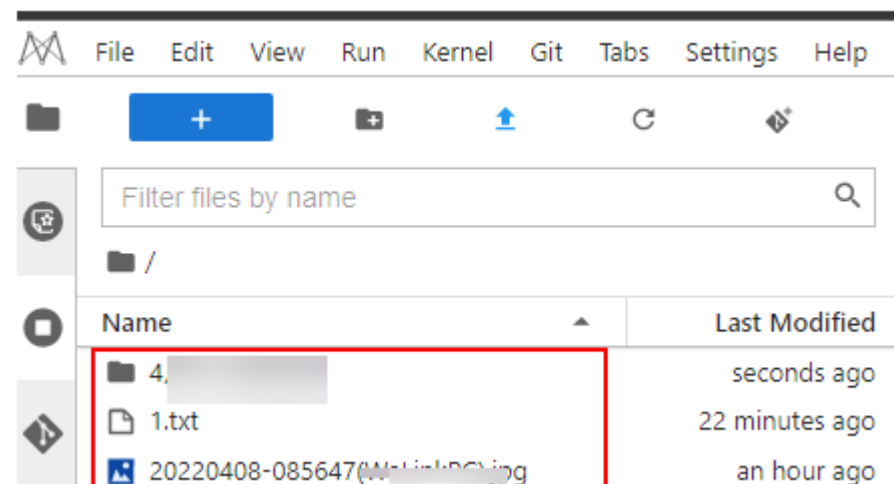


Figure 5-82 File downloaded to a notebook instance



Error Handling

If you download a file from OBS to your notebook instance and the system displays error message "Permission denied", perform the following operations for troubleshooting:

- Ensure that the target OBS bucket and notebook instance are in the same region. If they are in different regions, the access to OBS will be denied.
- In this case, ensure that the notebook account has the permission to read data in the OBS bucket.

For details, see [Incorrect OBS Path on ModelArts](#).

5.5.5.2 Cloning GitHub Open-Source Repository Files to JupyterLab

Files can be cloned from a GitHub open-source repository to JupyterLab.

1. Use JupyterLab to open a running notebook instance.
2. Click  in the navigation bar on the top of the JupyterLab window. In the

 displayed dialog box, click  on the left to go to the page for cloning files from a GitHub open-source repository.

Figure 5-83 File upload icon

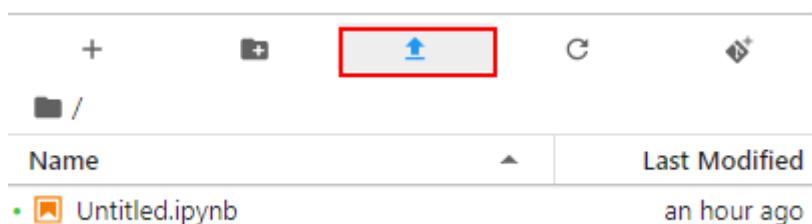
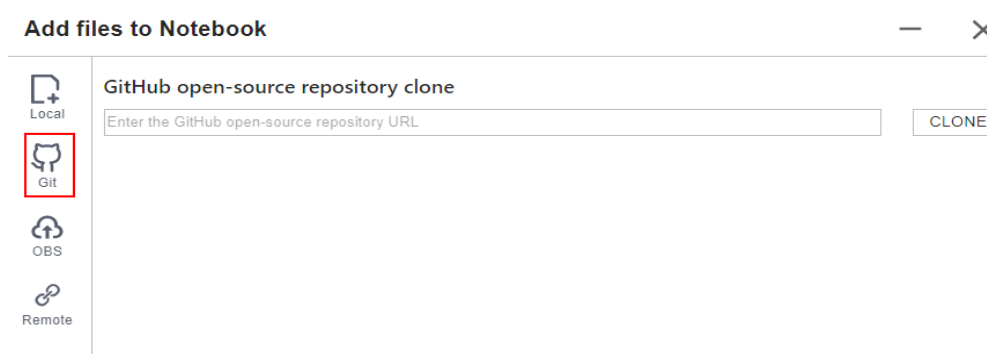
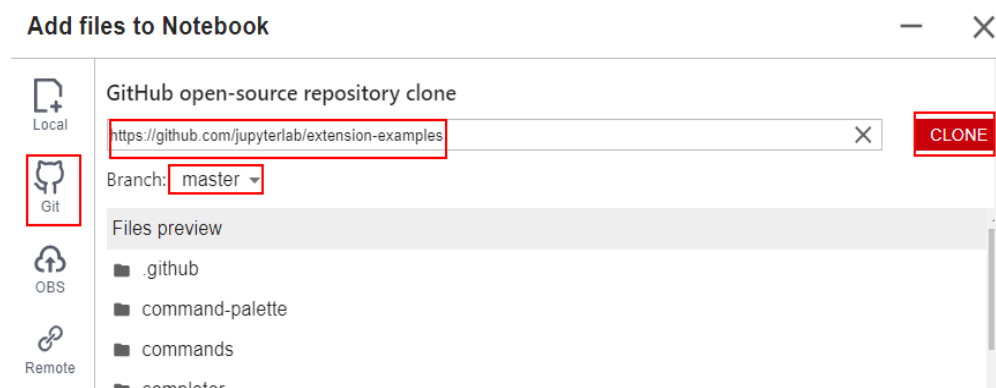


Figure 5-84 Page for cloning files from a GitHub open-source repository

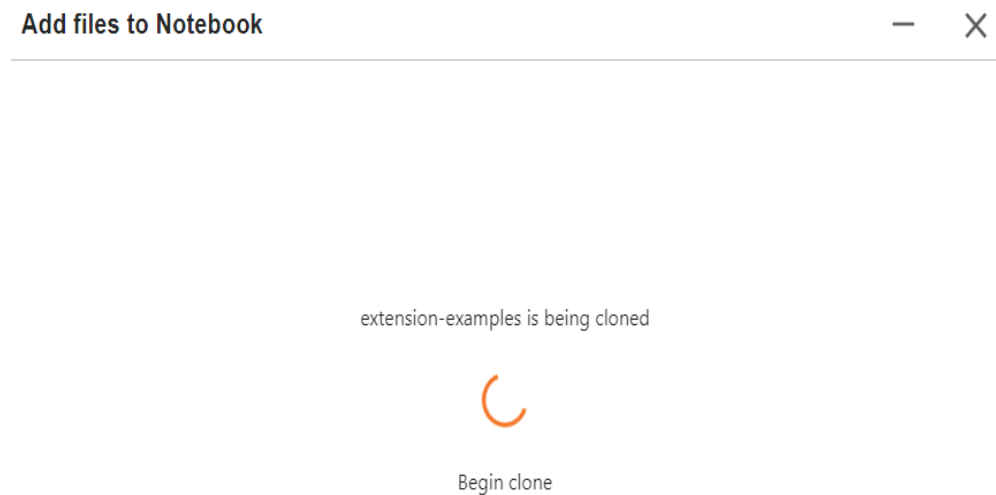


3. Enter a valid address of a GitHub open-source repository, select files from the displayed files and folders, and click **Clone**.

GitHub open-source repository address: <https://github.com/jupyterlab/extension-examples>

Figure 5-85 Entering a valid address of a GitHub open-source repository

4. View the clone process.

Figure 5-86 Process of cloning a repository

5. Complete the clone.

Error Handling

- Failing to clone the repository may be caused by network issues. In this case, run the **git clone https://github.com/jupyterlab/extension-examples.git** command on the **Terminal** page to test the network connectivity.
- If the repository already exists in the current directory of the notebook instance, the system displays a message indicating that the repository name already exists. In this case, you can overwrite the existing repository or click



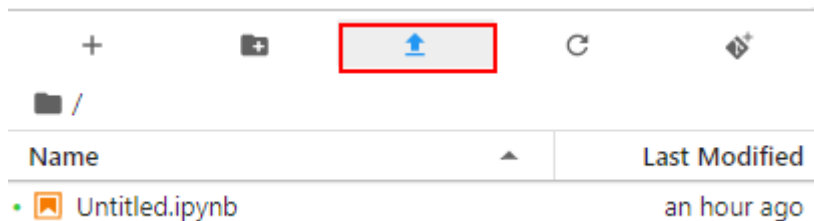
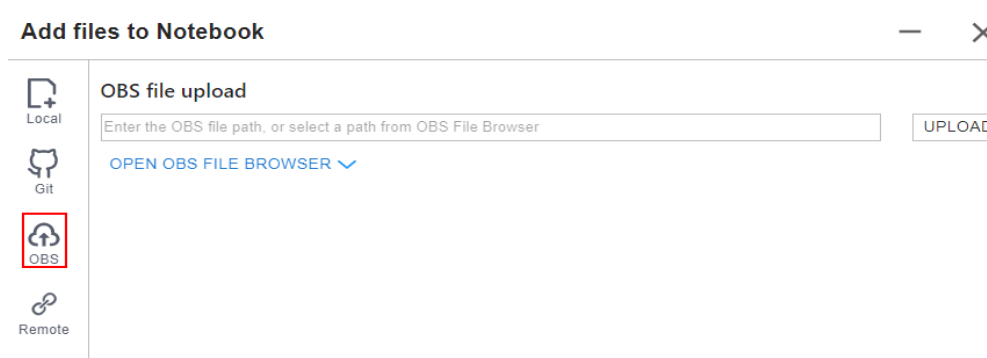
to cancel the cloning.

5.5.5.3 Uploading OBS Files to JupyterLab

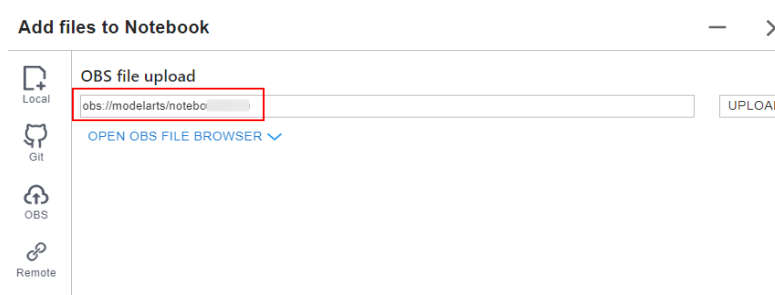
In JupyterLab, you can download files from OBS to a notebook instance. Ensure that the file is not larger than 10 GB. Otherwise, the upload will fail.

1. Use JupyterLab to open a running notebook instance.

- Click  in the navigation bar on the top of the JupyterLab window. In the displayed window, click  on the left to go to the OBS file upload page.

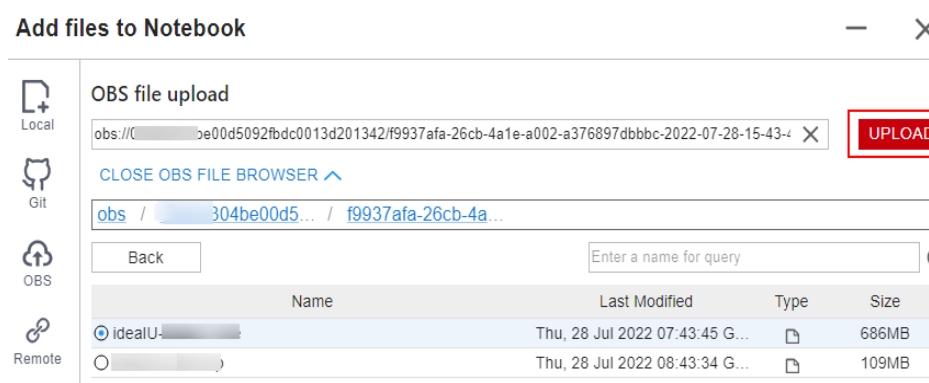
Figure 5-87 File upload icon**Figure 5-88** OBS file upload

- Set an OBS file path in either of the following ways:
 - Method 1: Enter a valid OBS file path in the text box and click **Upload**.

Figure 5-89 Entering a valid OBS file path**NOTE**

Enter an OBS file path instead of a folder path. Otherwise, the upload fails.

- Method 2: Open **OBS File Browser**, select an OBS file path, and click **Upload**.

Figure 5-90 Uploading an OBS File

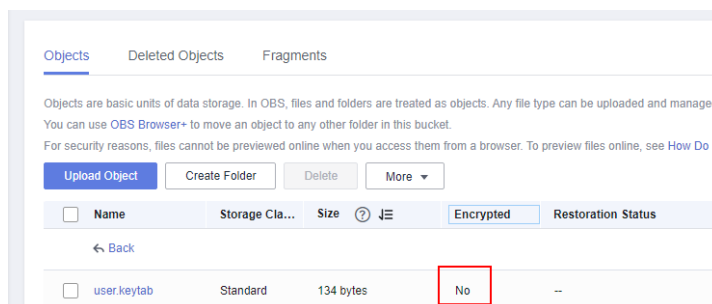
Error Handling

There are three typical scenarios in which uploading a file failed.

- **Scenario 1**

Possible causes:

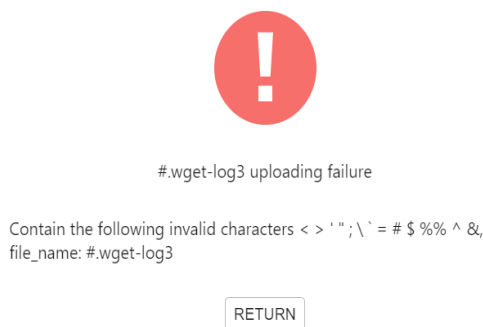
- The OBS path is set to a folder instead of a file path.
- The file in OBS is encrypted. In this case, go to the OBS console and ensure that the file is encrypted.



- The OBS bucket and notebook instance are not in the same region. Ensure that the OBS bucket to be read is in the same region as the notebook instance. You cannot access an OBS bucket in another region. For details, see [How Do I Check Whether ModelArts and an OBS Bucket Are in the Same Region?](#)
- The account does not have the permission to access the OBS bucket. In this case, ensure that the notebook account has the permission to read data in the OBS bucket. For details, see [Check Whether You Have Permission to Access the OBS Bucket.](#)
- The OBS file has been deleted. In this case, make sure that the OBS file to be uploaded is available.

- **Scenario 2**

Figure 5-91 File uploading failure

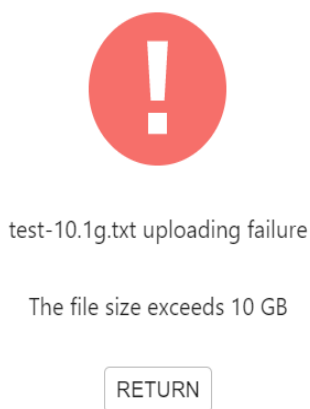


Possible causes:

The file name contains special characters such as <>"';\`=#\$%^&.

- **Scenario 3**

Figure 5-92 File uploading failure



Possible causes:

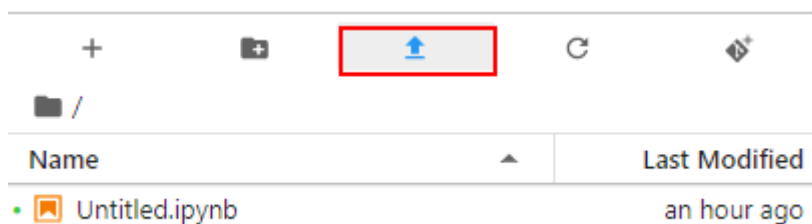
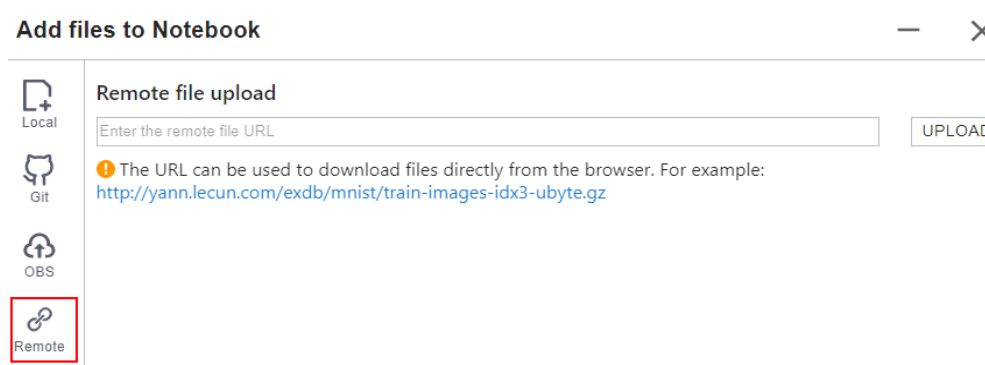
The uploaded file exceeded 50 GB.

5.5.5.4 Uploading Remote Files to JupyterLab

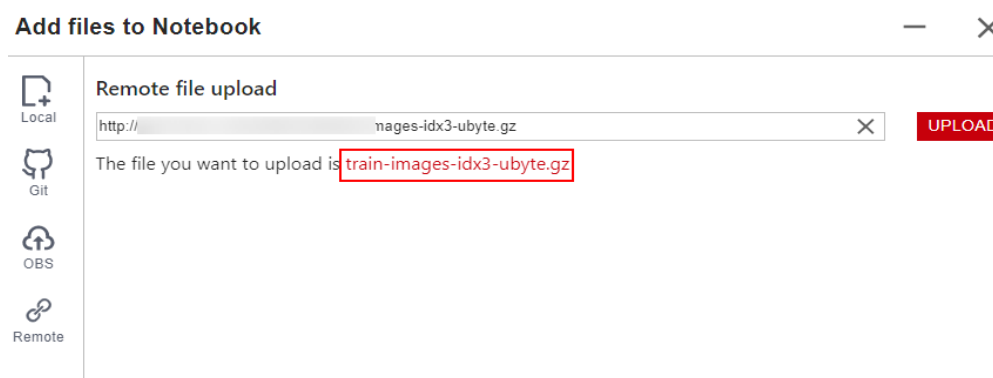
Files can be downloaded through remote file addresses to JupyterLab.

Method: Enter the URL of a remote file in the text box of a browser, and the file is directly downloaded.

1. Use JupyterLab to open a running notebook instance.
2. Click  in the navigation bar on the top of the JupyterLab window. In the displayed window, click  on the left to go to the remote file upload page.

Figure 5-93 File upload icon**Figure 5-94** Remote file upload page

3. Enter a valid remote file URL, and the system automatically identifies the file name. Then, click **Upload**.

Figure 5-95 Entering a valid remote file URL

Error Handling

Failing to upload the remote file may be caused by network issues. In this case, enter the URL of the remote file in the text box of a browser to check whether the file can be downloaded.

5.5.6 Downloading a File from JupyterLab to a Local PC

Files created in JupyterLab can be downloaded to a local path. For details about how to upload files to JupyterLab, see [Uploading Files to JupyterLab](#).

- If a file is less than or equal to 100 MB, directly download it from JupyterLab. For details, see [Downloading a File Less Than or Equal to 100 MB](#).

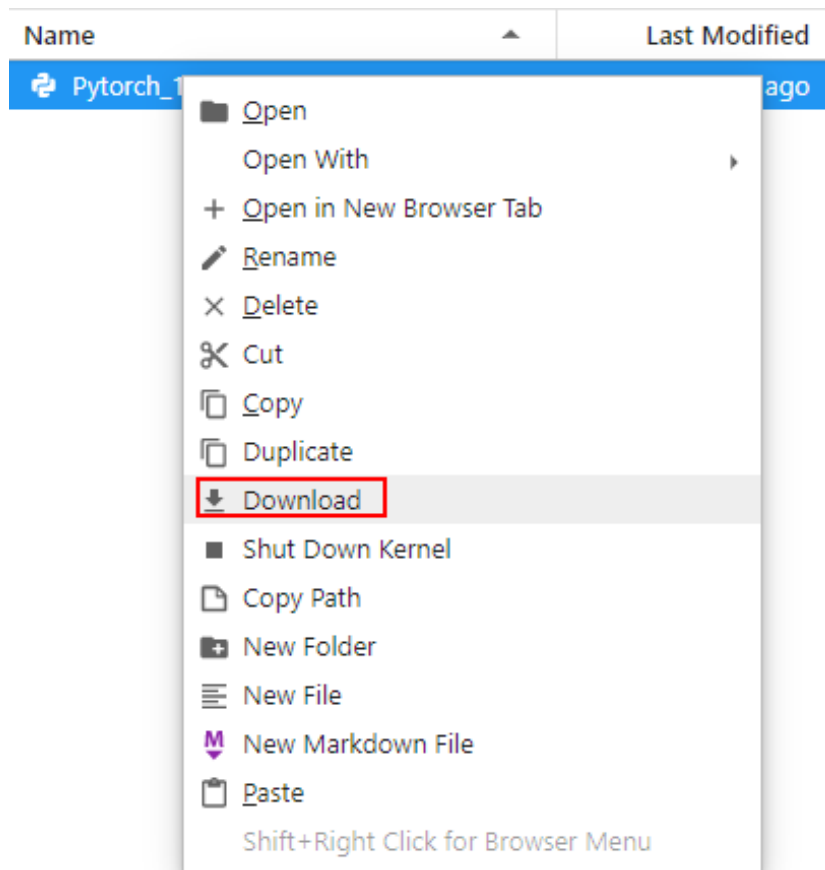
- If a file is larger than 100 MB, use OBS to transfer it to your local path. For details, see [Downloading a File Larger Than 100 MB](#).

Downloading a File Less Than or Equal to 100 MB

In the JupyterLab file list, right-click the file to be downloaded and choose **Download** from the shortcut menu.

The file is downloaded to your browser's downloads folder.

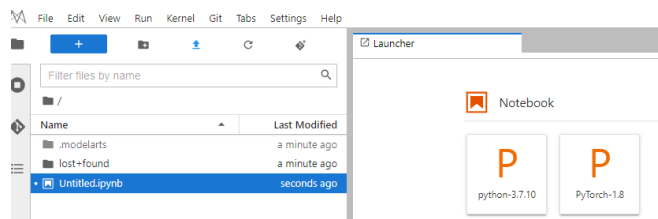
Figure 5-96 Downloading a file



Downloading a File Larger Than 100 MB

Use OBS to transfer the file from the target notebook instance to the local path. To do so, perform the following operations:

1. Open the Python runtime environment.
In the **Launcher** tab, click **python-3.7.10** under **Notebook**, as shown in the following figure.

Figure 5-97 Opening the Python runtime environment

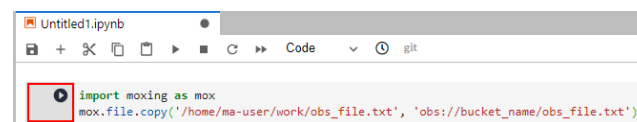
2. Use MoXing to upload the target file from the notebook instance to OBS.

The following shows the sample Python code for uploading the TXT file and compressed folder. **/home/ma-user/work/xxx** indicates the path for storing the file in the notebook instance. **obs://bucket_name/xxx** indicates the path for storing the file on OBS. Replace them with the actual paths.

- Upload the **obs_file.txt** file from the notebook instance to OBS.

Enter the following code, modify the path as required, and click  to run the code. If the TXT file exists in the OBS bucket, the upload is successful.

```
import moxing as mox
mox.file.copy('/home/ma-user/work/obs_file.txt', 'obs://bucket_name/obs_file.txt')
```

Figure 5-98 Running code example

- Upload the compressed folder **sub_dir_0** from the notebook instance to OBS.

Enter the following code, modify the path as required, and click  to run the code. If the folder exists in the OBS bucket, the upload is successful.

```
import moxing as mox
mox.file.copy_parallel('/home/ma-user/work/sub_dir_0', 'obs://bucket_name/sub_dir_0')
```

3. Use OBS or ModelArts SDK to download the file from OBS to the local path.

- Method 1: Use OBS to download the file.

Download **obs_file.txt** from OBS to the local path. If a large amount of data is to be downloaded, use OBS Browser+ to download. For details, see [Downloading an Object](#).

- Method 2: Use ModelArts SDK to download the file.

- i. [Download and install the SDK locally.](#)

- ii. [Authenticate sessions.](#)

- iii. [Download the file from OBS to the local path.](#) Example code is as follows:

```
from modelarts.session import Session

# Hardcoded or plaintext AK/SK is risky. For security, encrypt your AK/SK and store them
# in the configuration file or environment variables.
# In this example, the AK and SK are stored in environment variables for identity
# authentication. Before running this example, set environment variables
# HUAWEICLOUD_SDK_AK and HUAWEICLOUD_SDK_SK.
__AK = os.environ["HUAWEICLOUD_SDK_AK"]
```

```
__SK = os.environ["HUAWEICLOUD_SDK_SK"]  
# Decrypt the password if it is encrypted.  
session = Session(access_key=__AK,secret_key=__SK, project_id='***', region_name='***')  
  
session.download_data(bucket_path="/bucket_name/obs_file.txt",path="/home/user/  
obs_file.txt")
```

5.5.7 Using MindInsight Visualization Jobs in JupyterLab

ModelArts notebook supports MindInsight visualization jobs. In a development environment, use a small dataset to train and debug an algorithm. This is used to check algorithm convergence and detect training issues, facilitating debugging.

MindInsight visualizes information such as scalars, images, computational graphs, and model hyperparameters during training. It also provides functions such as training dashboard, model lineage, data lineage, and performance debugging, helping you train and debug models efficiently. MindInsight supports MindSpore training jobs. For more information about MindInsight, see [MindSpore official website](#).

MindSpore allows you to save data into the summary log file and obtain the data on the MindInsight GUI.

Prerequisites

When using MindSpore to edit a training script, add the code for collecting the summary record to the script to ensure that the summary file is generated in the training result.

For details, see [Collecting Summary Record](#).

Note

- To run a MindInsight training job in a development environment, start MindInsight and then the training process.
- Only one-card single-node training is supported.
- A running visualization job is not billed separately. When the target notebook instance is stopped, the billing stops.
- If the summary file is stored in OBS, OBS storage will be billed separately. After a job is complete, stop the notebook instance and clear OBS data to stop billing.

Creating a MindInsight Visualization Job in a Development Environment

[Step 1 Create a Development Environment and Access It Online](#)

[Step 2 Upload the Summary Data](#)

[Step 3 Start MindInsight](#)

[Step 4 View Visualized Data on the Training Dashboard](#)

Step 1 Create a Development Environment and Access It Online

Log in to ModelArts management console. In the navigation pane on the left, choose **Development Workspace** > **Notebook**, and create a development

environment instance using the MindSpore engine. After the instance is created, click **Open** in the **Operation** column of the instance to access it online.

Step 2 Upload the Summary Data

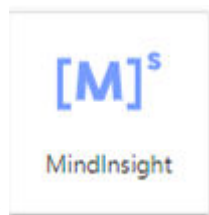
Summary data is required for MindInsight visualization in a development environment.

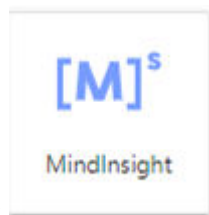
Upload the summary data to the `/home/ma-user/work/` directory in a development environment or store it in an OBS parallel file system.

- For details about how to upload the summary data to the notebook path `/home/ma-user/work/`, see [Uploading Files from a Local Path to JupyterLab](#).
- To store the summary data in an OBS parallel file system that is mounted to a notebook instance, upload the summary file generated during model training to the OBS parallel file system and ensure that the OBS parallel file system and ModelArts are in the same region. When MindInsight is started in a notebook instance, the notebook instance automatically reads the summary data from the mounted OBS parallel file system.

Step 3 Start MindInsight

Open MindInsight in JupyterLab.

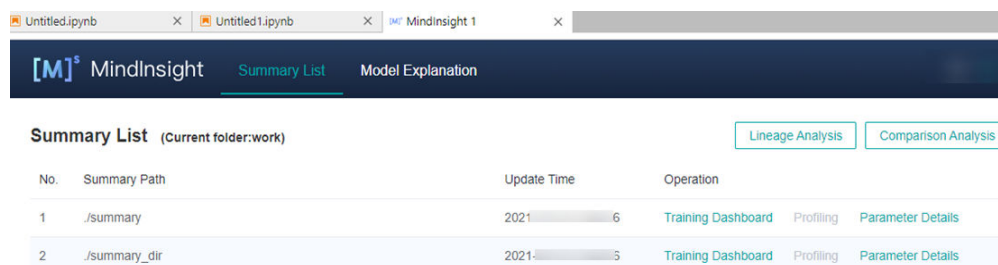


Click  to go to the MindInsight page.

Data is read from `/home/ma-user/work/` by default.

If there are two projects or more, select the target project to view its logs.

Figure 5-99 MindInsight page (2)



Step 4 View Visualized Data on the Training Dashboard

The training dashboard is important for MindInsight visualization. It allows visualization for scalars, parameter distribution, computational graphs, dataset graphs, images, and tensors.

Disabling MindInsight

Click . The MindInsight instance management page is displayed, which shows all started MindInsight instances. Click **SHUT DOWN** next to the target instance to stop it.

Figure 5-100 Stopping an instance



5.5.8 Using TensorBoard Visualization Jobs in JupyterLab

ModelArts supports TensorBoard for visualizing training jobs. TensorBoard is a visualization tool package of TensorFlow. It provides visualization functions and tools required for machine learning experiments.

TensorBoard effectively displays the computational graph of TensorFlow in the running process, the trend of all metrics in time, and the data used in the training. For more details about TensorBoard, see [TensorBoard official website](#).

TensorBoard visualization training jobs support only CPU and GPU flavors based on TensorFlow and PyTorch images. Select images and flavors based on the site requirements.

Prerequisites

When you write a training script, add the code for collecting the summary record to the script to ensure that the summary file is generated in the training result.

For details about how to add the code for collecting the summary record to a TensorFlow-powered training script, see [TensorFlow official website](#).

Precautions

- A running visualization job is not billed separately. When the target notebook instance is stopped, the billing stops.
- If the summary file is stored in OBS, you will be charged for the storage. After a job is complete, stop the notebook instance and clear OBS data to stop billing.

Process of Creating a TensorBoard Visualization Job in a Development Environment

Step 1 Create a Development Environment and Access It Online

Step 2 Upload the Summary Data

Step 3 Start TensorBoard

Step 4 View Visualized Data on the Training Dashboard

Step 1 Create a Development Environment and Access It Online

Log in to the ModelArts management console. In the navigation pane on the left, choose **Development Workspace > Notebook**. Create an instance using a TensorFlow or PyTorch image. After the instance is created, click **Open** in the **Operation** column of the instance to access it online.

TensorBoard visualization training jobs support only CPU and GPU flavors based on TensorFlow and PyTorch images. Select images and flavors based on the site requirements.

Step 2 Upload the Summary Data

Summary data is required for using TensorBoard visualization functions in DevEnviron.

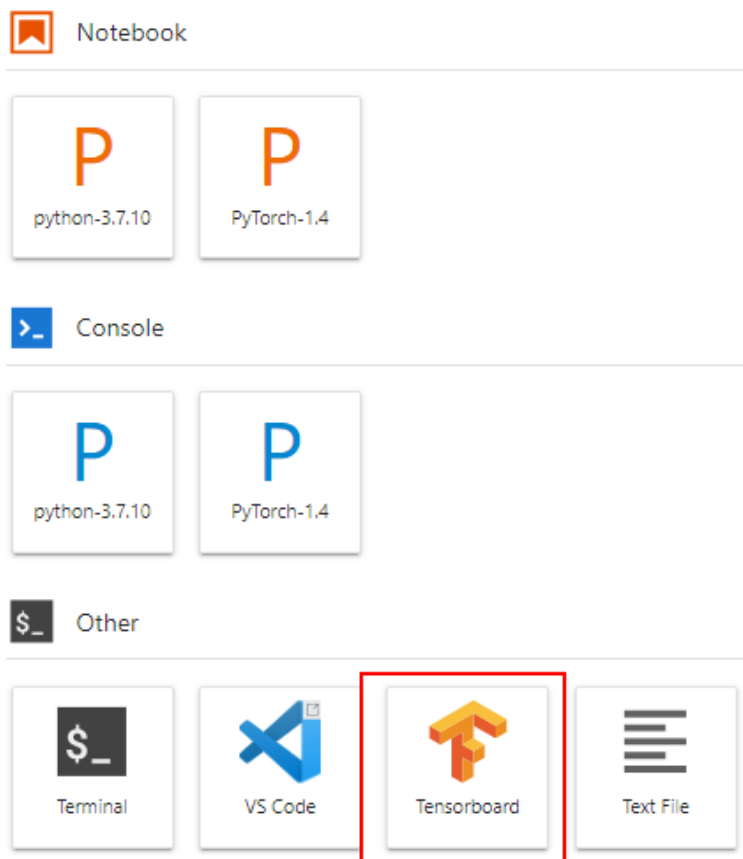
You can upload the summary data to the `/home/ma-user/work/` directory in the development environment or store it in the OBS parallel file system.

- For details about how to upload the summary data to the notebook path `/home/ma-user/work/`, see [Uploading Files from a Local Path to JupyterLab](#).
- To store the summary data in an OBS parallel file system that is mounted to a notebook instance, upload the summary file generated during model training to the OBS parallel file system and ensure that the OBS parallel file system and ModelArts are in the same region. When TensorBoard is started in a notebook instance, the notebook instance automatically mounts the OBS parallel file system directory and reads the summary data.

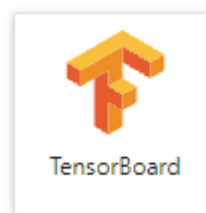
Step 3 Start TensorBoard

Choose a way you like to start TensorBoard in JupyterLab.

Figure 5-101 Starting TensorBoard in JupyterLab

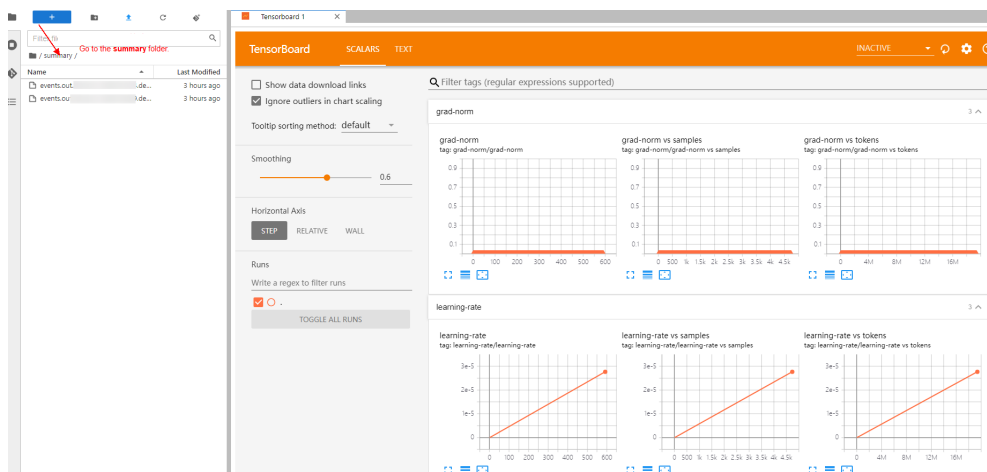


1. Open JupyterLab, in the navigation pane on the left, create the **summary** folder, and upload data to **/home/ma-user/work/summary**. The folder name must be **summary**.



2. Go to the **summary** folder and click TensorBoard page. See [Figure 5-102](#). to go to the

Figure 5-102 TensorBoard page (1)



Step 4 View Visualized Data on the Training Dashboard

For TensorBoard visualization, you need the training dashboard. It lets you visualize scalars, images, and computational graphs.

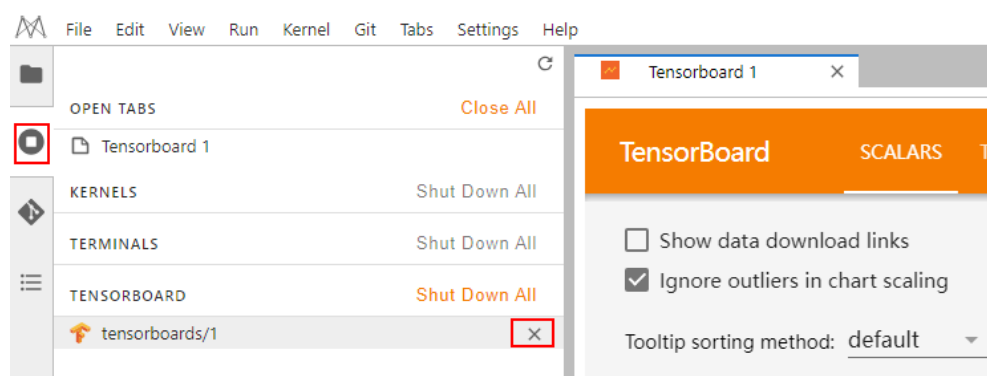
For more functions, see [Get started with TensorBoard](#).

Disabling TensorBoard

To stop a TensorBoard instance, use any of the following methods:

Click . The TensorBoard instance management page is displayed, which shows all started TensorBoard instances. Click **SHUT DOWN** next to an instance.

Figure 5-103 Clicking SHUT DOWN to stop an instance



5.6 Using Notebook Instances Remotely Through PyCharm

5.6.1 Connecting to a Notebook Instance Through PyCharm Toolkit

AI developers use PyCharm to develop algorithms or models. ModelArts provides the PyCharm Toolkit plug-in to help AI developers quickly submit locally developed code to the ModelArts training environment. With PyCharm Toolkit, developers can quickly remotely access notebook instances, upload code, submit training jobs, and obtain training logs for local display so that they can better focus on local code development.

This section introduces how to connect to a notebook instance through PyCharm Toolkit.

Constraints

- Currently, only versions between 2019.2 and 2023.2 (including 2019.2 and 2023.2) are supported, including the community edition and professional edition.
- PyCharm of only the professional edition can be used to access notebook instances.
- You can use a community or professional edition of PyCharm Toolkit to submit training jobs. The latest version of PyCharm Toolkit can be used only to submit training jobs of the new version.
- PyCharm Toolkit supports only PyCharm of the Windows version.

Table 5-23 Toolkit functions of the latest version

Supported Function	Description	Reference
Remote SSH connection	The notebook development environment can be accessed through remote SSH.	Connecting to a Notebook Instance Through PyCharm Toolkit
OBS-based upload and download	Local files or folders can be uploaded to OBS and files or folders can be downloaded from OBS to a local directory.	Uploading Data to a Notebook Instance Through PyCharm

Prerequisites

PyCharm professional edition of a version between 2019.2 and 2023.2 (including 2019.2 and 2023.2) has been installed on the local PC. Remote SSH applies only to the PyCharm professional edition. [Download PyCharm](#) and install it.

Step 1 Downloading and Installing PyCharm Toolkit

In PyCharm, choose **File > Settings > Plugins**, search for **ModelArts** in Marketplace, and click **Install**.

Step2 Creating a Notebook Instance

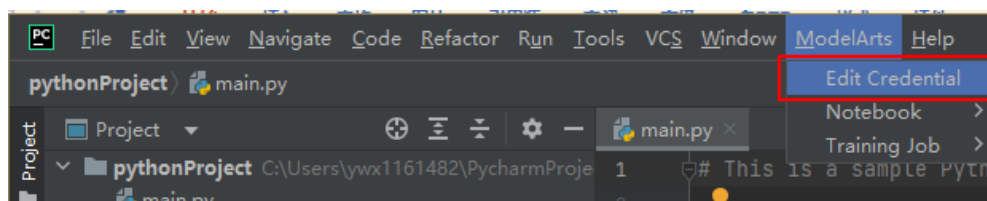
Create a notebook instance with remote SSH enabled and whitelist configured. Ensure that the instance is running. For details, see [Creating a Notebook Instance \(Default Page\)](#).

Step 3 Logging In to the Plug-in

To use the AK/SK pair for login authentication, perform the following steps:

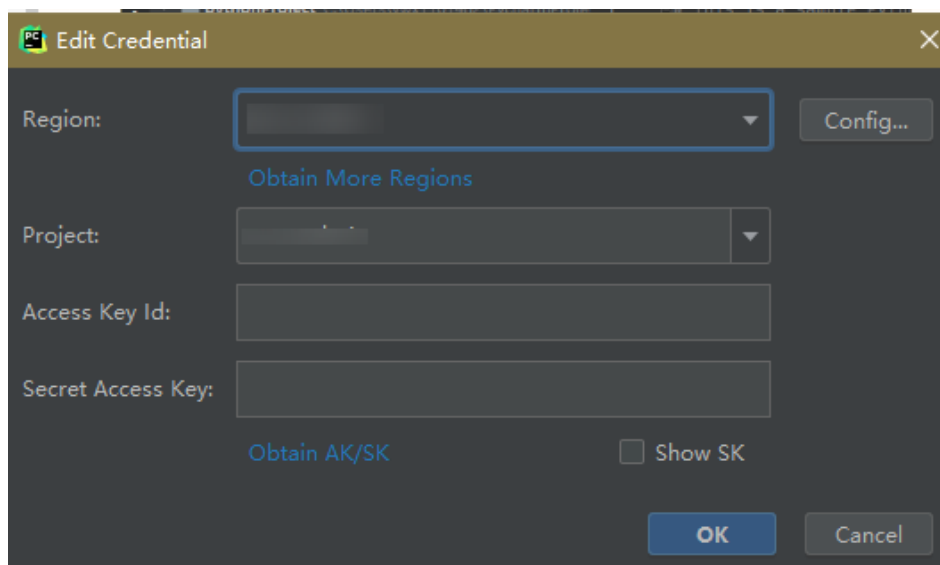
1. Open PyCharm with Toolkit installed. Choose **ModelArts > Edit Credential** from the menu bar.

Figure 5-104 Edit Credential



NOTE

- If **ModelArts > Edit Credential** is not displayed on the menu bar, the PyCharm version may be too high. PyCharm Toolkit is not adapted to PyCharm versions later than 2023.2. Download the PyCharm professional edition of a version between 2019.2 and 2023.2 (including 2019.2 and 2023.2).
2. In the displayed dialog box, select the region where ModelArts is located, enter the AK and SK, and click **OK**. For details about how to obtain the AK/SK, see [How Do I Obtain an Access Key?](#)
 - **Region:** Select a region from the drop-down list. It must be the same as the region of the ModelArts console.
 - **Project:** After the region is selected, the project is automatically filled.
 - **Access Key ID:** Enter the AK.
 - **Secret Access Key:** Enter the SK.

Figure 5-105 Entering the region and access keys

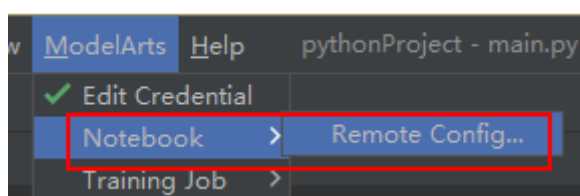
3. View the verification result.

In the **Event Log** area, if information similar to the following is displayed, the access key has been successfully added:

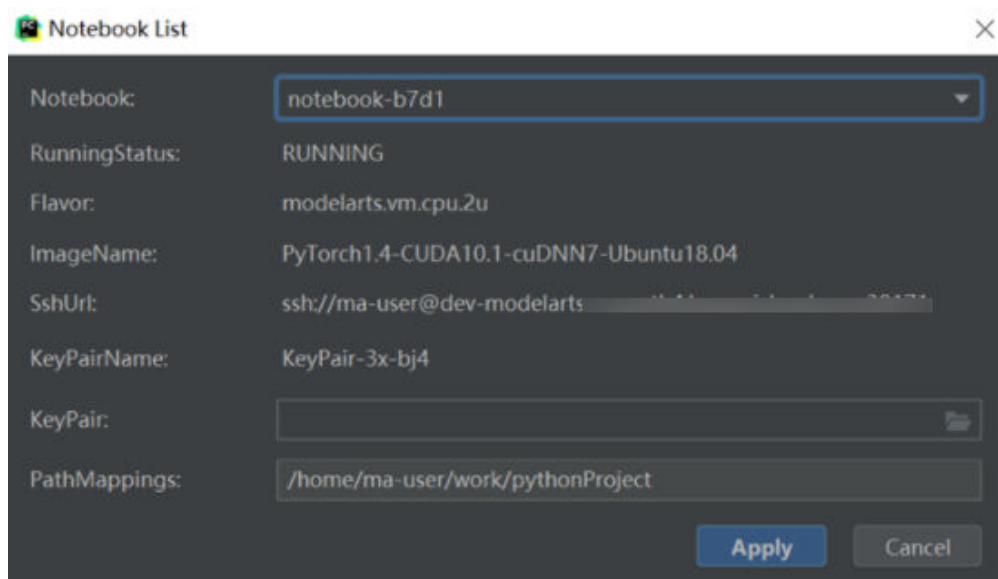
16:01Validate Credential Success: The HUAWEI CLOUDcredential is valid.

Step 4 Automatically Configuring PyCharm Toolkit

1. In the local PyCharm development environment, choose **ModelArts > Notebook > Remote Config...**, and configure PyCharm Toolkit.

Figure 5-106 Remotely connecting to PyCharm Toolkit

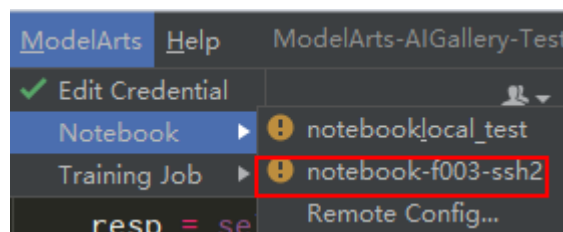
2. Choose the target instance from the drop-down list, where all notebook instances with remote SSH enabled under the account are displayed.

Figure 5-107 Notebook list

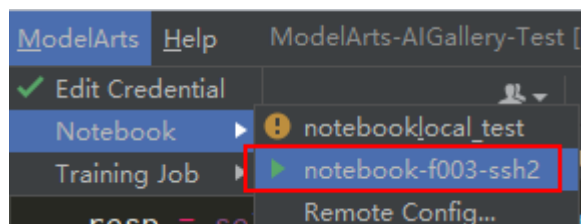
- **KeyPair:** Select the locally stored key pair of the notebook instance for authentication. The key pair created during the notebook instance creation is saved in your browser's default downloads folder.
 - **PathMappings:** Synchronization directory for the local IDE project and notebook, which defaults to `/home/ma-user/work/Project name` and is adjustable.
3. Click **Apply**. After the configuration is complete, restart the IDE for the configuration to take effect.
- After the restart, it takes about 20 minutes to update the Python interpreter for the first time.

Step 5 Accessing a Notebook Instance Through PyCharm Toolkit

Click the notebook instance name and connect it to the local IDE as prompted. The connection is kept for 4 hours by default.

Figure 5-108 Starting the connection

To interrupt the connection, click the notebook name and disconnect it from the local IDE as prompted.

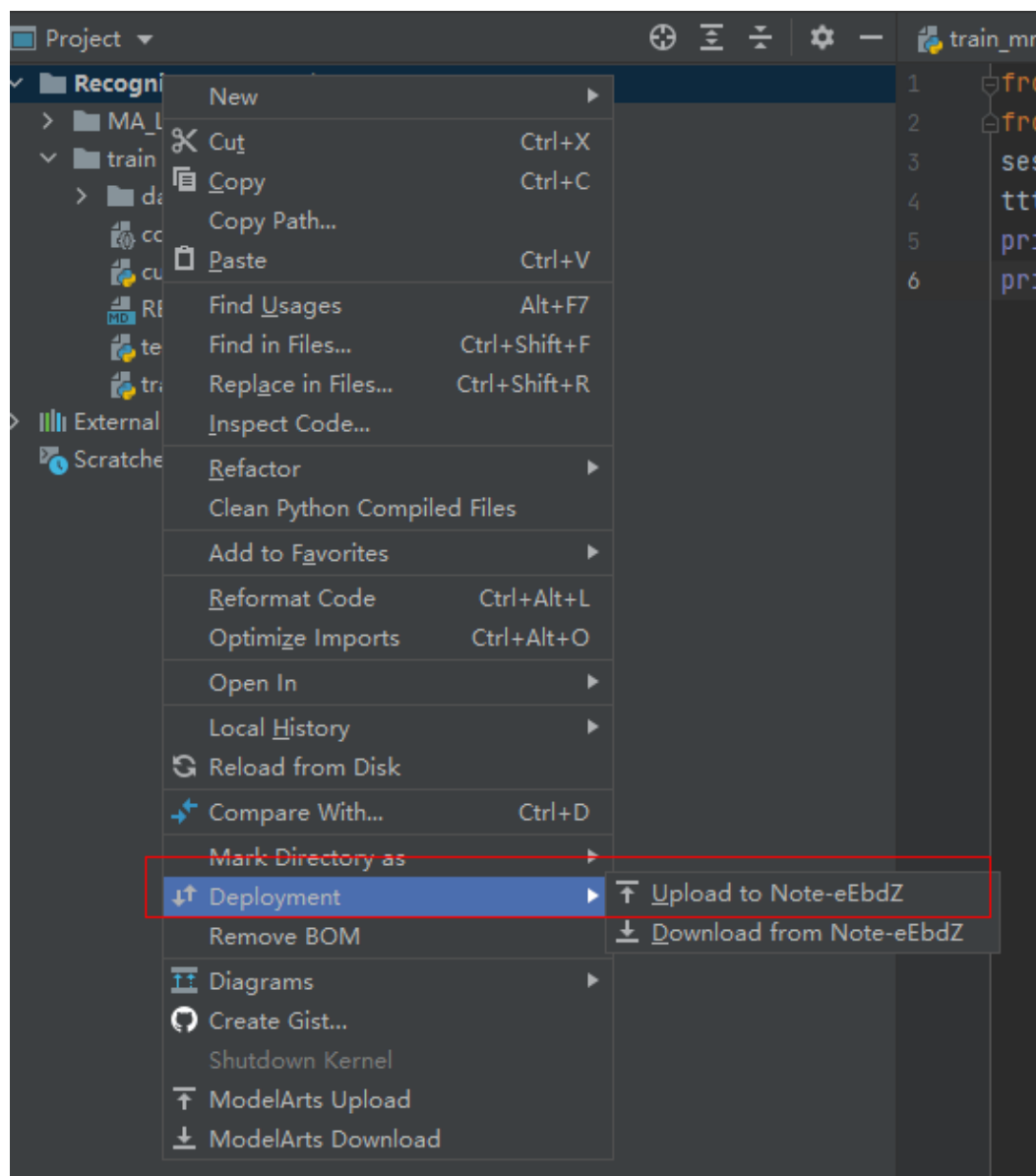
Figure 5-109 Interrupting the connection

Step 6 Uploading Local Files to a Notebook Instance

Code in a local file can be copied to the local IDE, which will automatically synchronize the code to the in-cloud development environment.

Initial synchronization

In the **Project** directory of the local IDE, right-click **Deployment** and choose **Upload to Notebook name** from the shortcut menu to upload the local project file to the specified notebook instance.

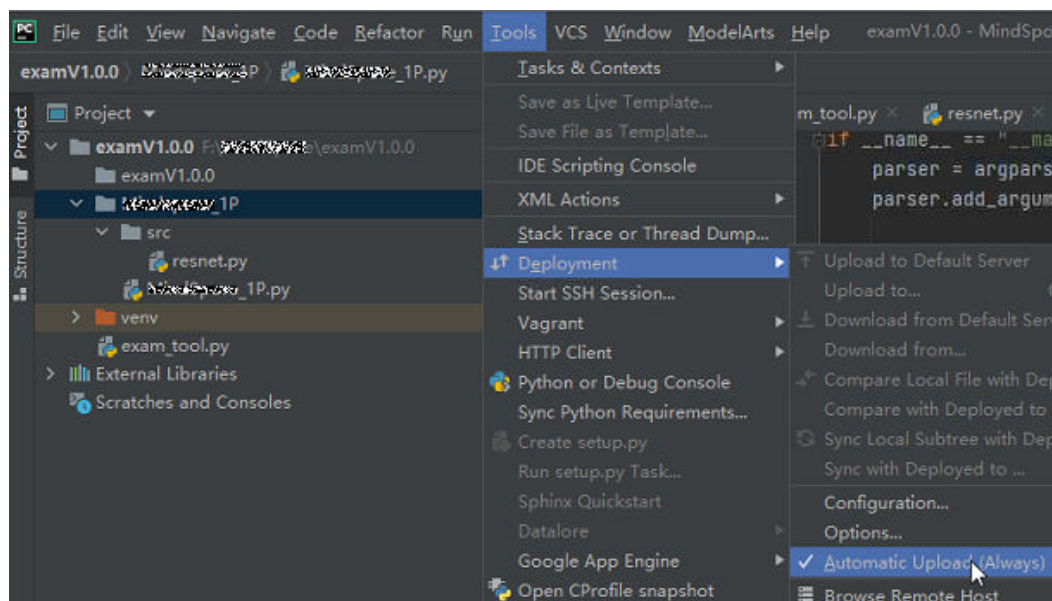
Figure 5-110 Synchronizing local data to a notebook instance

Follow-up synchronization

After modifying the code, press **Ctrl+S** to save it. The local IDE will automatically synchronize the modification to the specified notebook instance.

After PyCharm Toolkit is installed, **Automatic Upload** is automatically enabled in the local IDE for automatically uploading the files in the local directory to the target notebook instance. If **Automatic Upload** is not enabled, enable it by referring to the following figure.

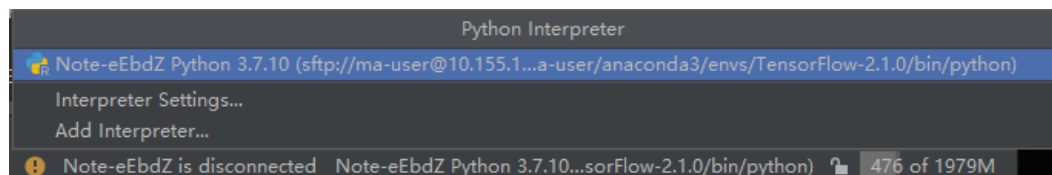
Figure 5-111 Enabling Automatic Upload



Step 7 Remotely Debugging the Code

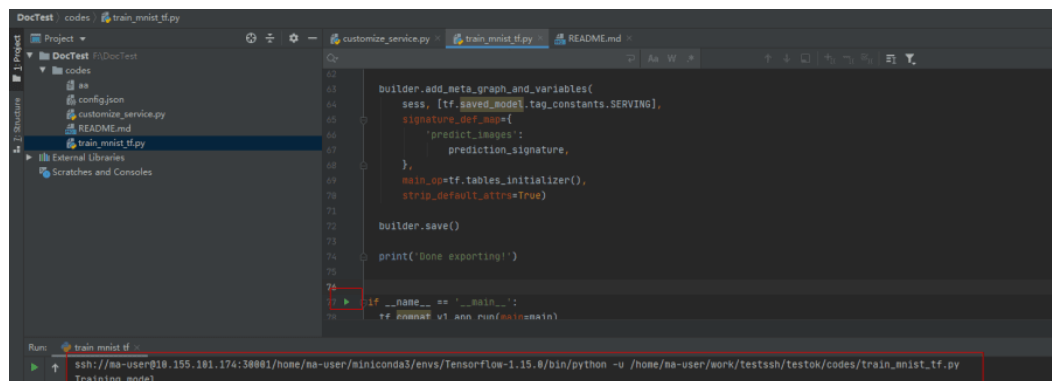
Click **Interpreter** in the lower right corner of the local IDE and select a notebook Python interpreter.

Figure 5-112 Selecting a Python interpreter

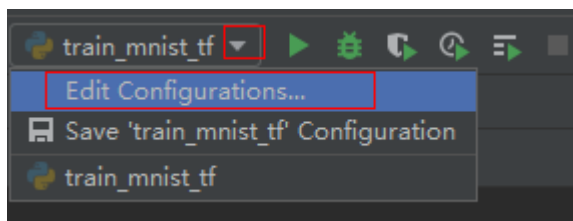


Run the code in the notebook instance. The logs are displayed locally.

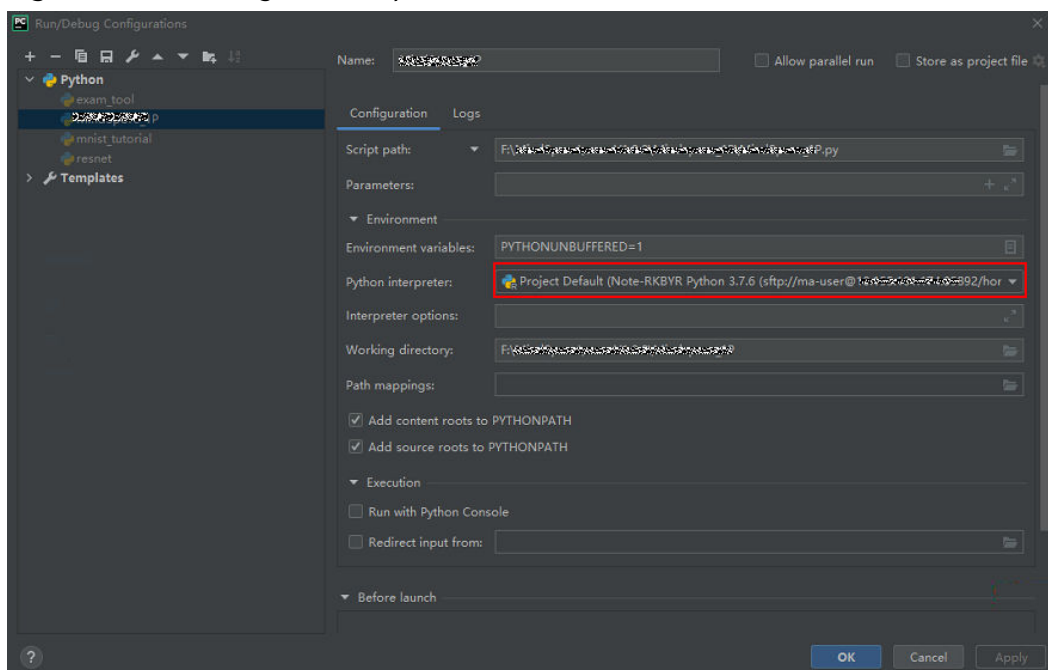
Figure 5-113 Runtime logs



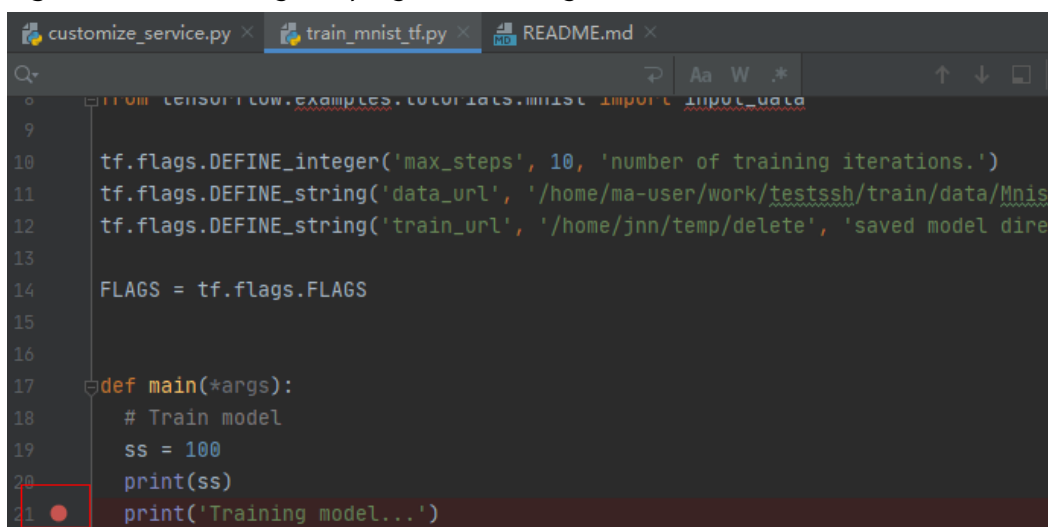
Click **Run/Debug Configurations** in the upper right corner of the local IDE to set runtime parameters.

Figure 5-114 Setting runtime parameters (1)

Select the Python interpreter that remotely connects to the target notebook instance.

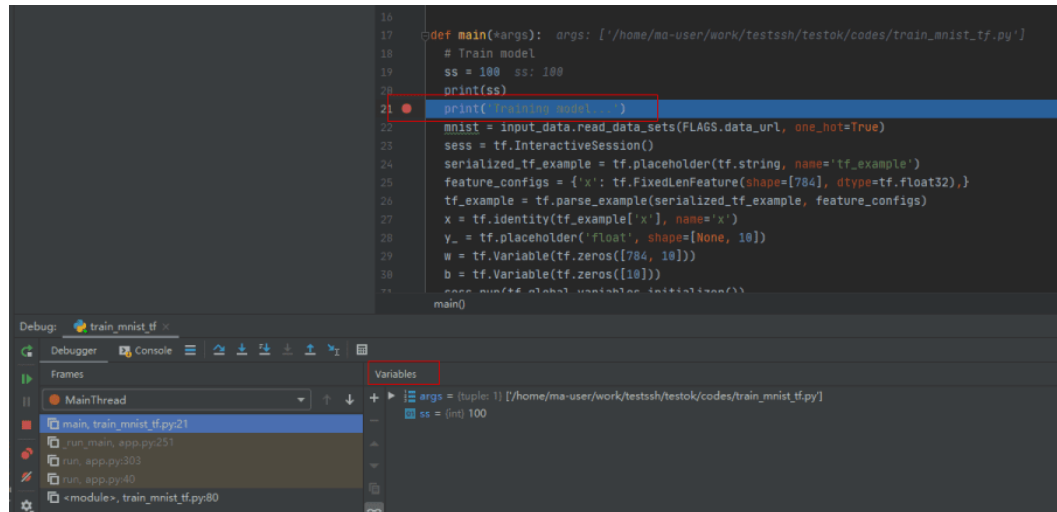
Figure 5-115 Setting runtime parameters (2)

To debug code, set breakpoints and run the program in debug mode.

Figure 5-116 Running the program in debug mode

In debug mode, the code execution is suspended in the specified line, and you can obtain variable values.

Figure 5-117 Viewing variable values in debug mode



5.6.2 Manually Connecting to a Notebook Instance Through PyCharm

A local IDE supports PyCharm and VS Code. You can use PyCharm or VS Code to remotely connect the local IDE to the target notebook instance on ModelArts for running and debugging code.

This section describes how to use PyCharm to access a notebook instance.

Prerequisites

- PyCharm professional 2019.2 or later has been installed locally. The PyCharm professional edition is available because remote SSH applies only to the professional edition.
- A notebook instance has been created with remote SSH enabled. Ensure that the instance is running. For details, see [Creating a Notebook Instance \(Default Page\)](#).
- The address and port number of the development environment are available. To obtain this information, go to the notebook instance details page.

Figure 5-118 Instance details page

Address	ssh://ma-user@dev-modelarts- [redacted] .com 32651
Authentication	KeyPair-9a64  Access address of the development environment Port number

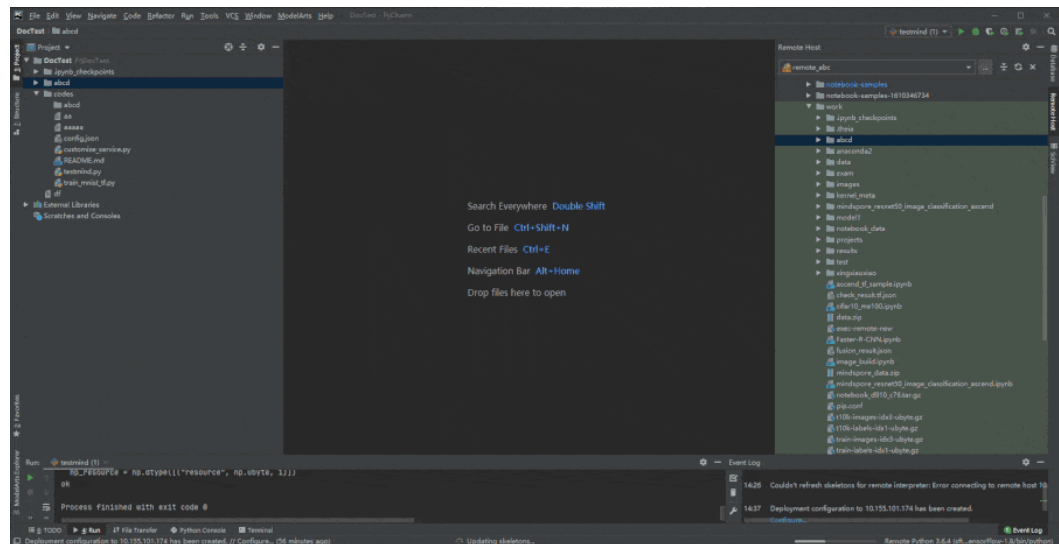
- The key pair is available.

A key pair is automatically downloaded after you create it. Securely store your key pair. If an existing key pair is lost, create a new one.

Step 1 Configure SSH

1. In your local PyCharm development environment, choose **File > Settings > Tools > SSH Configurations** and click **+** to add an SSH configuration.
 - **Host:** address for accessing the cloud development environment. Obtain the address on the page providing detailed information of the target notebook instance .
 - **Port:** port number for accessing the cloud development environment. Obtain the port number on the page providing detailed information of the target notebook instance.
 - **User name:** consistently set to **ma-user**.
 - **Authentication type:** key pair
 - **Private key file:** locally stored private key file of the remote cloud development environment. It is the key pair file automatically downloaded when you created the notebook instance.
2. Click  to rename the connection. Then, click **OK**.
3. After the configuration is complete, click **Test Connection** to test the connectivity.
4. Select **Yes**. If "Successfully connected" is displayed, the network is accessible. Then, click **OK**.
5. Click **OK** at the bottom to save the configuration.

Figure 5-119 Configuring SSH



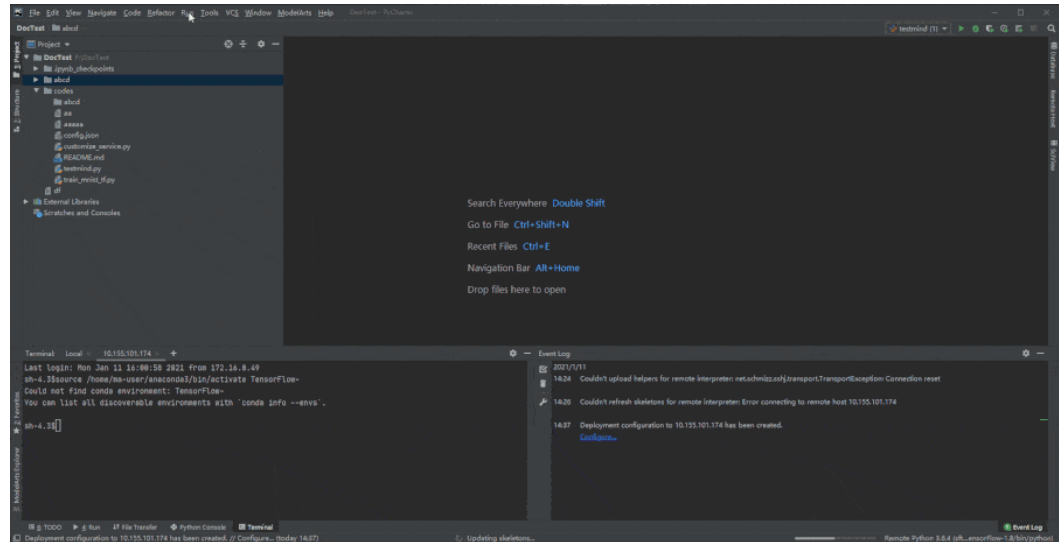
Step 2 Obtain the Path to the Virtual Environment Built in the Development Environment

1. Choose **Tools > Start SSH Session** to access the cloud development environment.
2. Run the following command to view the Python virtual environments built in the current environment in the **README** file in **/home/ma-user/**:

```
cat /home/ma-user/README
```

3. Run the **source** command to switch to a specific Python environment.
4. Run **which python** to obtain the Python path and copy it for configuring the Python interpreter on the cloud.

Figure 5-120 Obtaining the path to the virtual environment built in the development environment



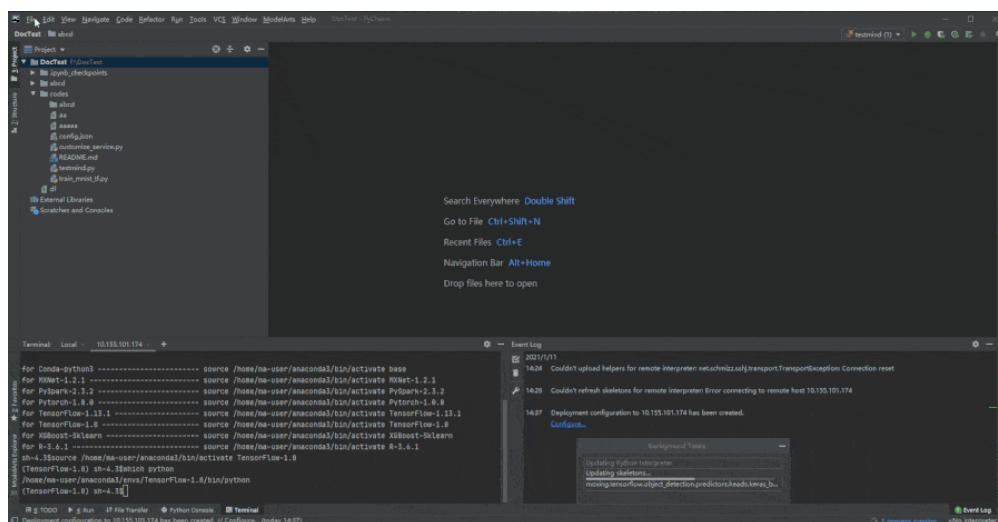
Step 3 Configure a Python Interpreter

1. Choose **File > Settings > Project: *Python project* > Python Interpreter**. Then, click  and **Add** to add an interpreter.
2. Select **Existing server configuration**, choose the SSH configuration from the drop-down list, and click **Next**.
3. Configure the Python interpreter.
 - **Interpreter:** Enter the Python path copied in step 1, for example, **/home/ma-user/anaconda3/envs/Pytorch-1.0.0/bin/python**.
If the path is **~/anaconda3/envs/Pytorch-1.0.0/bin/python**, replace **~** with **/home/ma-user**.
 - **Sync folders:** Set this parameter to a directory in the cloud development environment for synchronizing local project directory files. A directory in **/home/ma-user** is recommended, for example, **/home/ma-user/work/projects**, because other directories may be prohibited from accessing.
4. Click **!** on the right and select **Automatically upload** so that the locally modified file can be automatically uploaded to the container.
5. Click **Finish**.

The local project file has been automatically uploaded to the cloud environment. Each time a local file is modified, the modification is automatically synchronized to the cloud environment.

In the lower right corner, the current interpreter is displayed as a remote interpreter.

Figure 5-121 Configuring a Python interpreter



Step 4 Install the Dependent Library for the Cloud Environment

After accessing the development environment, you can use different virtual environments, such as TensorFlow and PyTorch. However, in actual development, you need to install dependency packages. Then, you can access the environment through the terminal to perform operations.

Choose **Tools > Start SSH Session** and select the configured development environment. Run the **pip install** command to install the required dependency packages.

```
Terminal: Local x 10.155.101.174 x +
Last login: Wed Dec 30 12:46:18 2020 from 10.155.101.174
sh-4.4$cat /home/ma-user/README
Please use one of following command to start the specified framework environment.

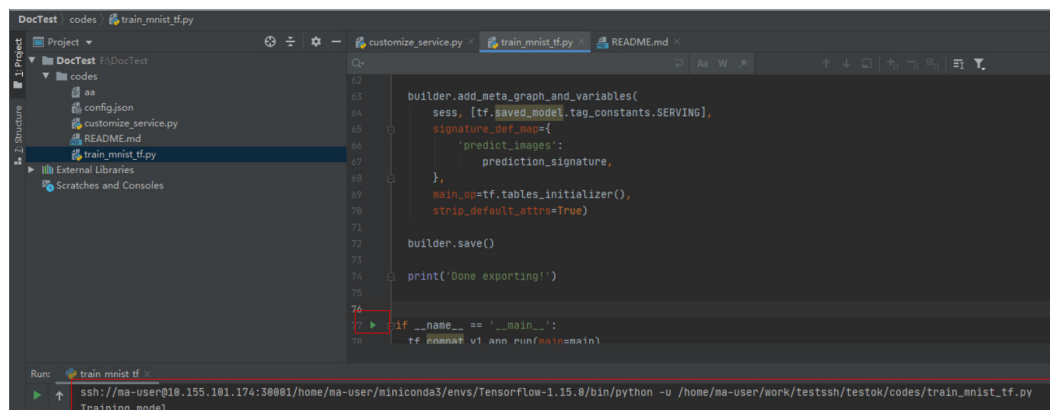
for Mindspore-1.0.1-python3.7-aarch64 ----- source /home/ma-user/miniconda3/bin/activate Mindspore-1.0.1-python3.7-aarch64
for Tensorflow-1.15.0 ----- source /home/ma-user/miniconda3/bin/activate Tensorflow-1.15.0
sh-4.4$source /home/ma-user/miniconda3/bin/activate Tensorflow-1.15.0
(Tensorflow-1.15.0) sh-4.4$which python
~/miniconda3/envs/Tensorflow-1.15.0/bin/python
(Tensorflow-1.15.0) sh-4.4$pwd
/home/ma-user
(Tensorflow-1.15.0) sh-4.4$pip install spacy
```

Step 5 Debug Code in the Development Environment

You have accessed the cloud development environment. Then, you can write, debug, and run the code in the local PyCharm. The code is actually executed in the cloud development environment, and the Ascend AI resources on the cloud are used. In this way, you compile and modify code locally and run the code in the cloud.

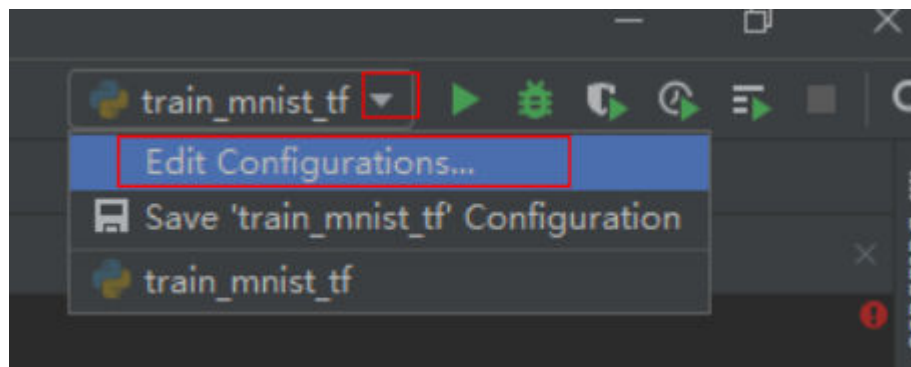
Run the code in the local IDE. The logs can be displayed locally.

Figure 5-122 Debugging code



Click **Run/Debug Configurations** in the upper right corner of the local IDE to set runtime parameters.

Figure 5-123 Setting runtime parameters



To debug code, set breakpoints and run the program in debug mode.

Figure 5-124 Code breakpoint

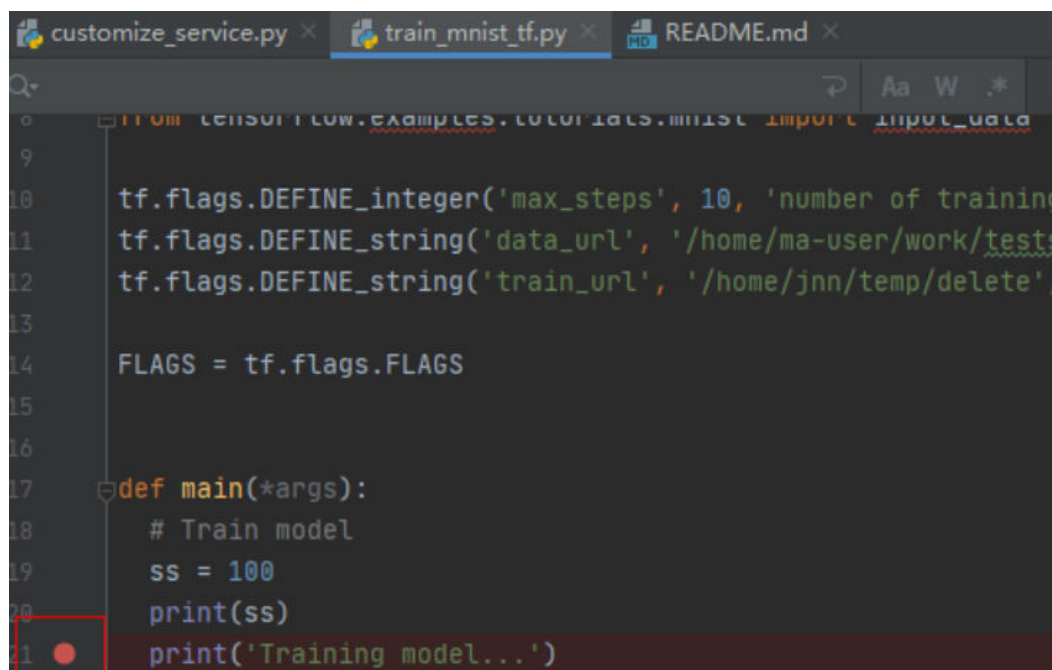
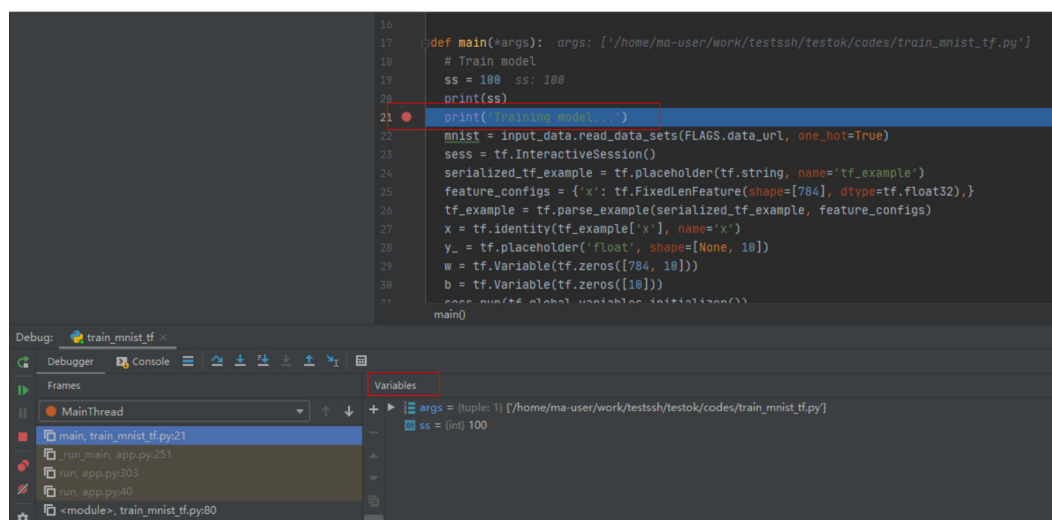


Figure 5-125 Debugging in debug mode



In debug mode, the code execution is suspended in the specified line, and you can obtain variable values.

Figure 5-126 Debug mode



Before debugging code in debug mode, ensure that the local code is the same as the cloud code. If they are different, the line where a breakpoint is added locally may be different from the line of the cloud code, leading to errors.

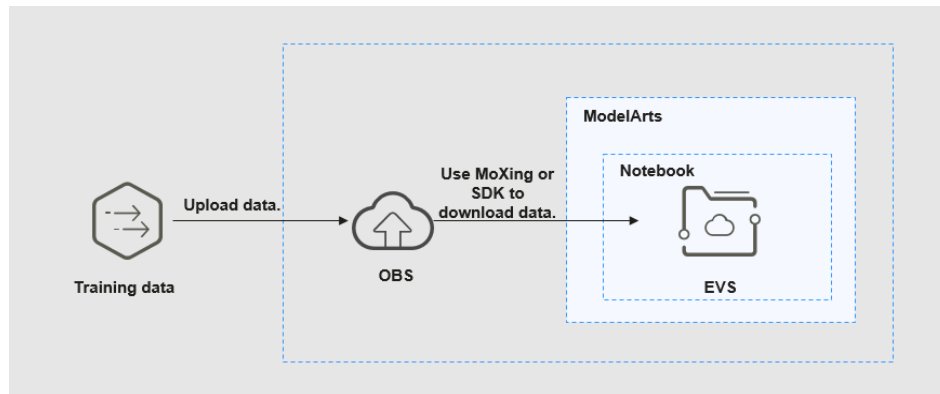
When configuring a Python interpreter in the cloud development environment, select **Automatically upload** so that any local file modification can be automatically uploaded to the cloud. If you do not select **Automatically upload**, manually upload the directory or code after you modify the local code. For details, see [Step 6 Uploading Local Files to a Notebook Instance](#).

5.6.3 Uploading Data to a Notebook Instance Through PyCharm

If the data is less than or equal to 500 MB, directly copy the data to the local IDE.

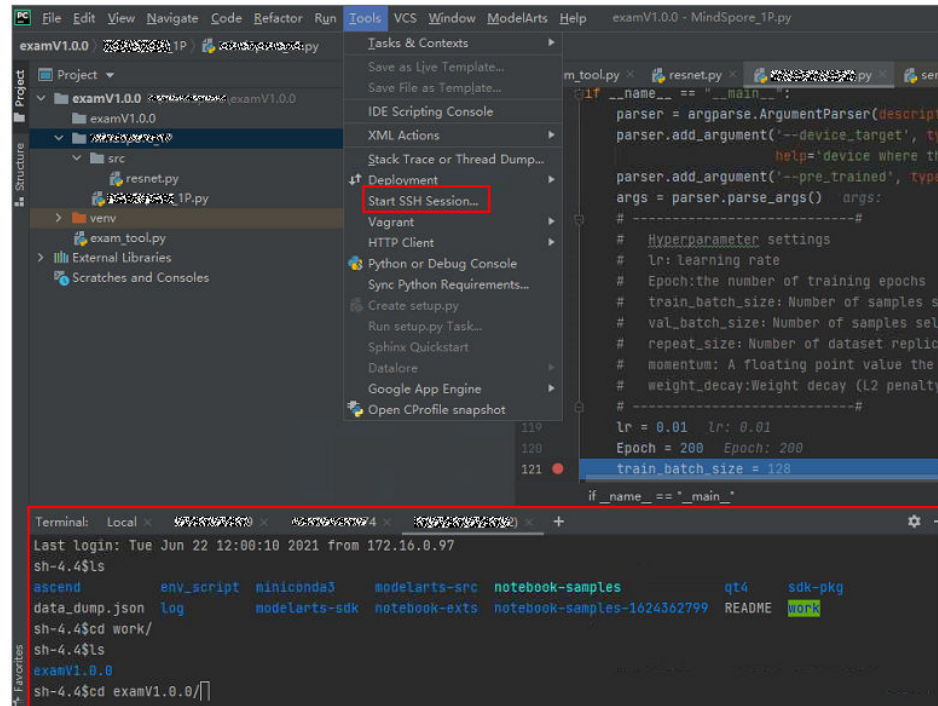
If the data is larger than 500 MB, upload it to OBS, and then download it to the notebook instance.

Figure 5-127 Uploading data to a notebook Instance through OBS



1. Upload data to OBS. For details, see [Uploading an Object](#).
2. Call the `mox.file.copy_parallel` MoXing API provided by ModelArts in the terminal of the local IDE to transfer data from OBS to the notebook instance.
 - a. Open terminal in PyCharm. The operations in Visual Studio Code (VS Code) are similar.

Figure 5-128 Enabling the terminal in PyCharm



- b. The following shows how to use MoXing in the terminal of the local IDE to upload files from OBS to a notebook instance:

```
# Manually access the development environment.
cat /home/ma-user/README
# Select the source environment.
source /home/ma-user/miniconda3/bin/activate MindSpore-python3.7-aarch64
# Enter python and press Enter to enter the Python environment.
python
# Use MoXing for access.
import moxing as mox
# Download a folder from OBS to EVS.
mox.file.copy_parallel('obs://bucket_name/sub_dir_0', '/tmp/sub_dir_0')
```

5.7 Using Notebook Instances Remotely Through VS Code

5.7.1 Connecting to a Notebook Instance Through VS Code

VS Code is a typical code editor that supports multiple programming languages and development environments. You can connect to and use Jupyter Notebook through VS Code.

After creating a notebook instance with remote SSH enabled, you can use VS Code to access the development environment in any of the following ways:

- **Connecting to a Notebook Instance Through VS Code Toolkit**
In this mode, log in to the ModelArts VS Code Toolkit plug-in and use it to connect to an instance.
- **Manually Connecting to a Notebook Instance Through VS Code**

In this mode, use the VS Code Remote-SSH plug-in to configure connection information and connect to an instance.

Installing VS Code

Install VS Code first.

- **Download URL:**

Download address: https://code.visualstudio.com/updates/v1_85

Figure 5-129 VS Code download URL

November 2023 (version 1.85)

Update 1.85.1: The update addresses these [issues](#).

Update 1.85.2: The update addresses these [issues](#).

Downloads: Windows: [x64](#) [Arm64](#) | Mac: [Universal Intel silicon](#) | Linux: [deb](#) [rpm](#) [tarball](#) [Arm snap](#)

- **VS Code version requirements:**

You are advised to use VS Code 1.85.2 or the latest version for remote connection.

- **VS Code installation guide:**

In Windows, double-click the installation package to complete the installation.

In Linux, run the command **sudo dpkg -i code_1.85.2-1705561292_amd64.deb** to install VS Code.

 NOTE

Linux system users must install VS Code as a non-root user.

5.7.2 Connecting to a Notebook Instance Through VS Code Toolkit

This section describes how to use the ModelArts VS Code Toolkit plug-in to remotely connect to a notebook instance.

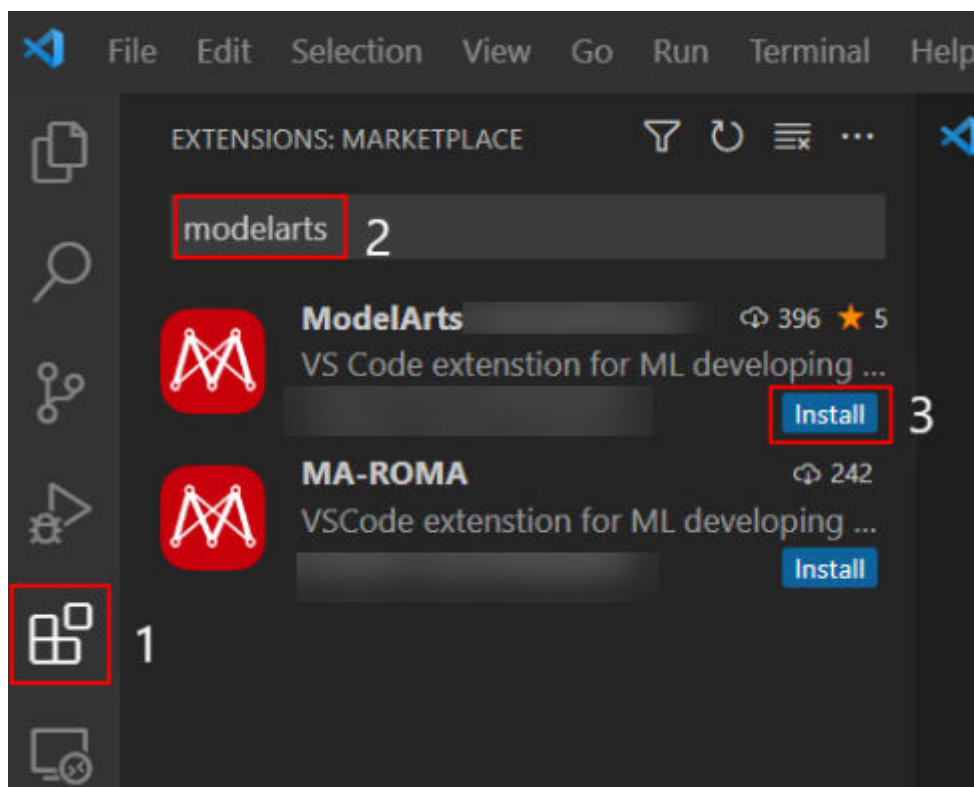
Prerequisites

You have downloaded and installed VS Code. For details, see [Connecting to a Notebook Instance Through VS Code](#).

Step 1 Install the VS Code Plug-in

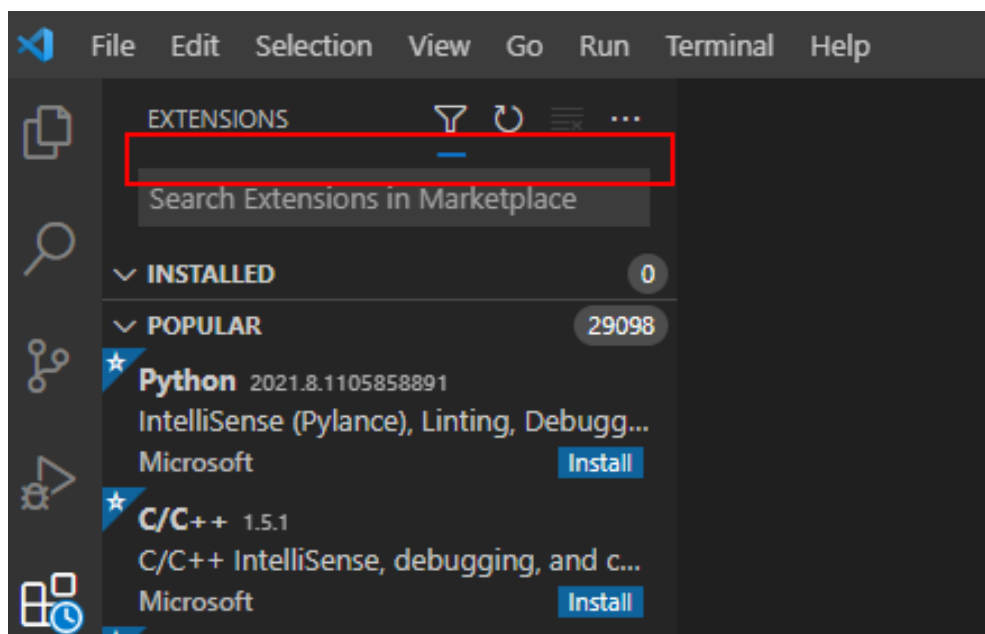
1. Search for **ModelArts-HuaweiCloud** in the **EXTENSIONS** text box and click **Install**.

Figure 5-130 Installing the VS Code plug-in

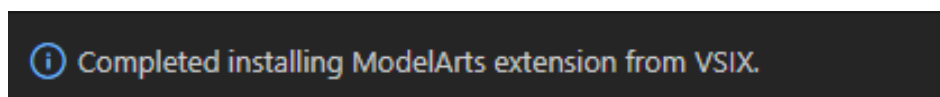
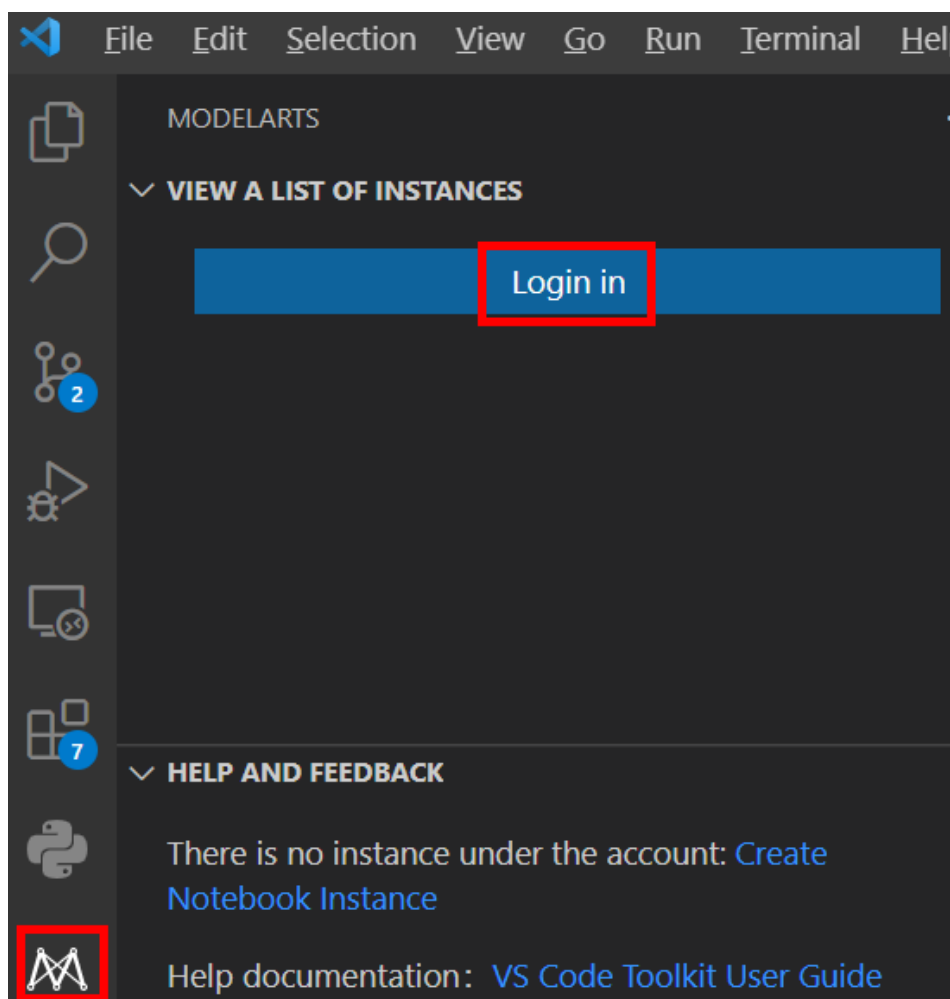


2. Wait for about 1 to 2 minutes.

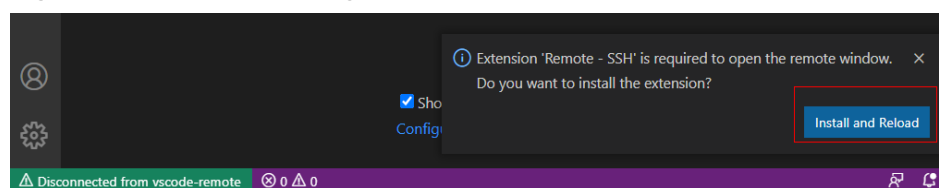
Figure 5-131 Installation process



3. After the installation is complete, check the message displayed in the lower right corner. If the ModelArts icon  and remote SSH icon  are displayed in the navigation pane on the left, the VS Code plug-in is installed.

Figure 5-132 Installation completion message**Figure 5-133** Installation completed

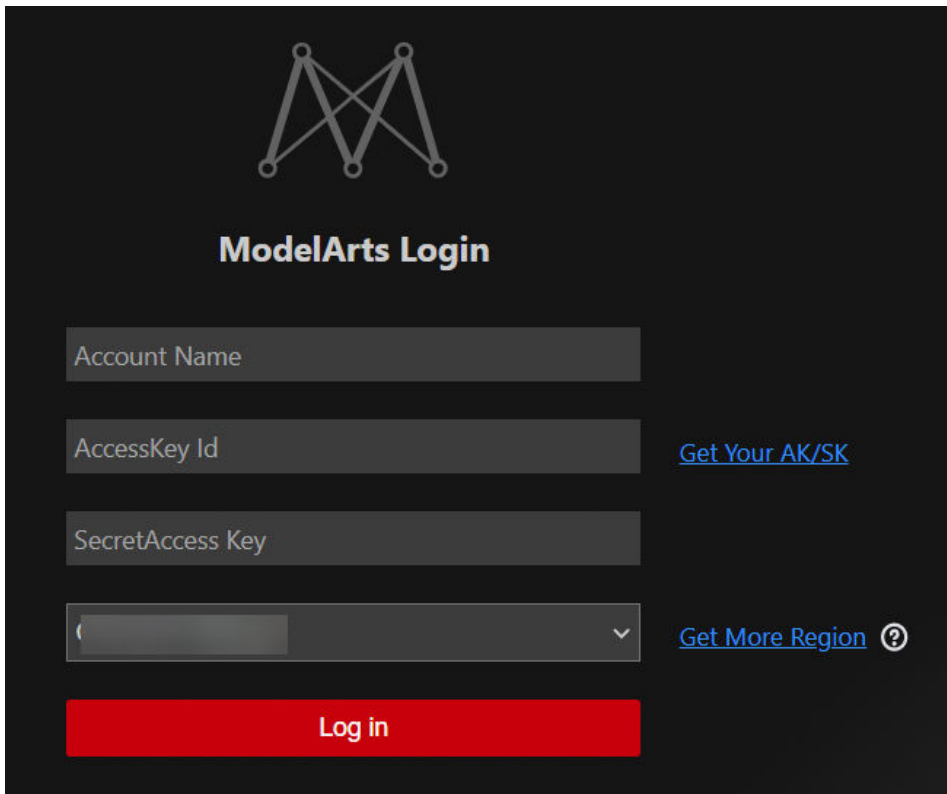
Network issues may cause an installation failure. If this occurs, proceed with follow-up operations. After 1 in **Step 4 Access the Notebook Instance** is performed, the system will automatically display a dialog box shown in the following figure. In this case, click **Install and Reload**.

Figure 5-134 Reconnecting remote SSH

Step 2 Log In to the VS Code Plug-in

1. In the local VS Code development environment, click  and **User Settings**, and configure the login information.

Figure 5-135 Logging in to the plug-in

The image shows a 'ModelArts Login' dialog box with a dark background. At the top is a logo consisting of three interconnected nodes. Below the logo, the text 'ModelArts Login' is displayed. There are four input fields: 'Account Name', 'AccessKey Id', 'SecretAccess Key', and a region dropdown menu. To the right of the 'AccessKey Id' field is a link 'Get Your AK/SK'. To the right of the region dropdown is a link 'Get More Region' followed by a question mark icon. At the bottom is a red 'Log in' button.

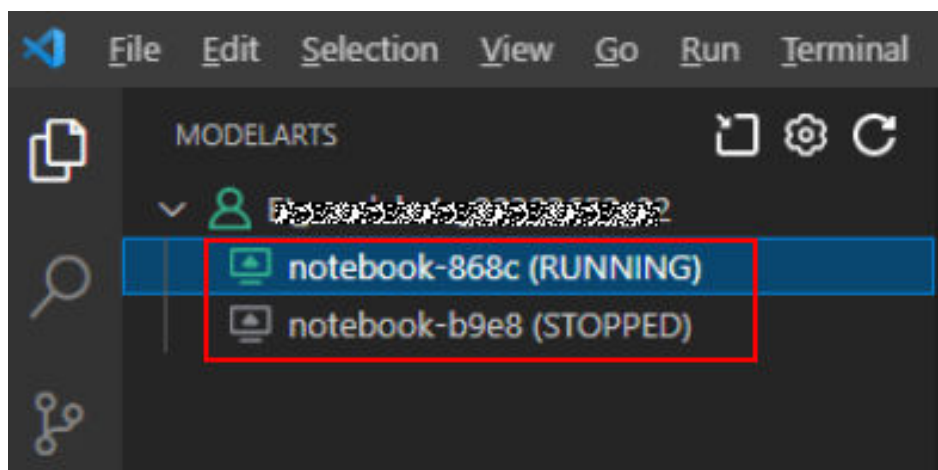
Enter the login information and click **Log in**.

- **Name:** Custom username, which is displayed only on the VS Code page and is not associated with any Huawei Cloud account.
 - **AK and SK:** Access key pair. To create a key pair, choose **My Credentials > API Credentials > Access Keys**, and click **Create Access Key**. For details, see [How Do I Obtain an Access Key?](#)
 - **Region:** must be the same as that of the notebook instance to be remotely connected. Otherwise, the connection will fail.
2. After the login, check the notebook instance list.

NOTE

The list displays only notebook instances in the default workspace on the ModelArts console.

Figure 5-136 Login succeeded



Step 3 Create a Notebook Instance

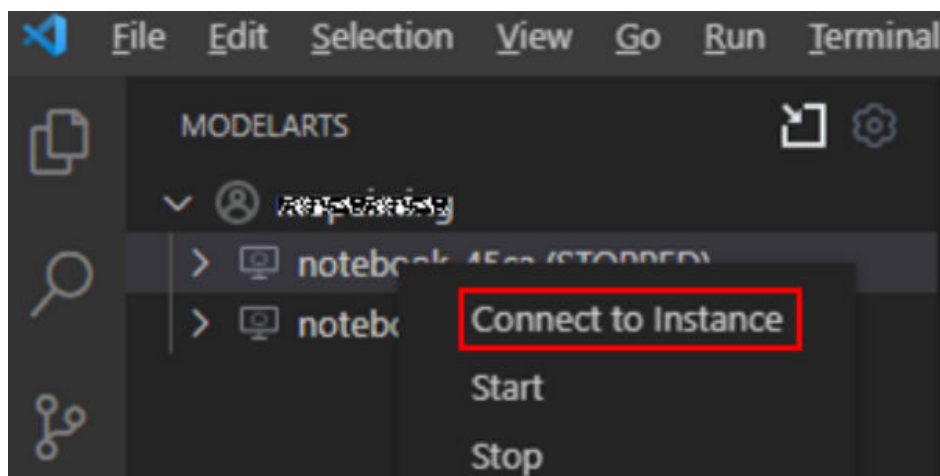
⚠ CAUTION

- Create a notebook instance with remote SSH enabled, and download the key file to either of the following directories based on your OS:
Windows: `C:\Users\{{user}}`
macOS/Linux: `Users/{{user}}`
- A key pair is automatically downloaded after you create it. Securely store your key pair. If an existing key pair is lost, create a new one.

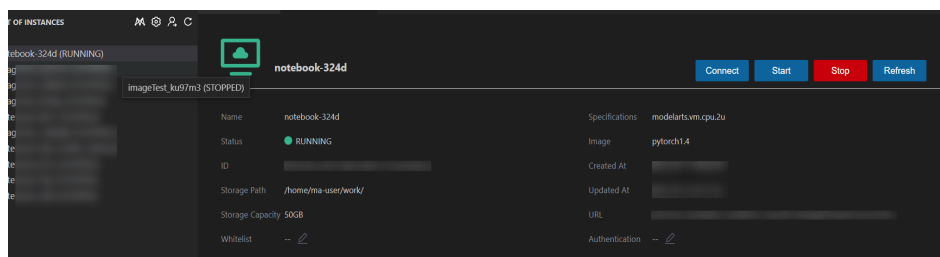
Create a notebook instance with remote SSH enabled. For details, see [Creating a Notebook Instance \(Default Page\)](#).

Step 4 Access the Notebook Instance

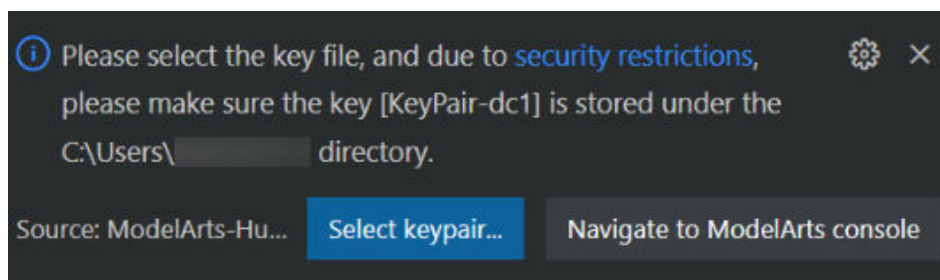
1. In the local VS Code development environment, right-click the instance name and choose **Connect to Instance** from the shortcut menu to start and connect to the notebook instance.
The notebook instance can either be running or stopped. If it is stopped, the VS Code plug-in starts the instance and then connects to it.

Figure 5-137 Connecting to a notebook instance

Alternatively, click the instance name. On the instance details page, click **Connect**. Then, the system automatically starts and connects to the notebook instance.

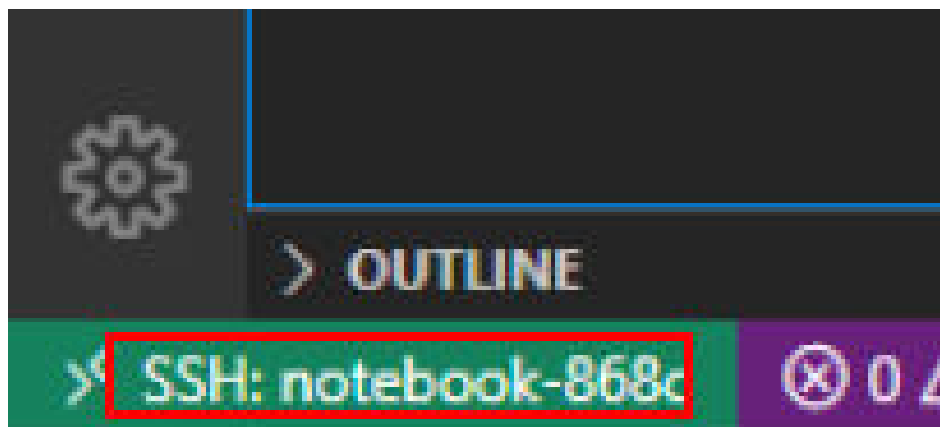
Figure 5-138 Viewing details about a notebook instance

2. When you connect to a notebook instance for the first time, the system prompts you in the lower right corner to configure the key file. In this case, select the local .pem key file and click **OK**.

Figure 5-139 Configuring the key file

3. Wait for about 1 to 2 minutes until the notebook instance is accessed. After information similar to the following is displayed in the lower left corner of the VS Code environment, the connection is succeeded.

Figure 5-140 Connection succeeded



Debugging Code Remotely

1. On the VS Code page, upload local code to the cloud development environment.
 - a. Choose **File > OpenFolder**, select the path to be opened, and click **OK**.

Figure 5-141 Open Folder

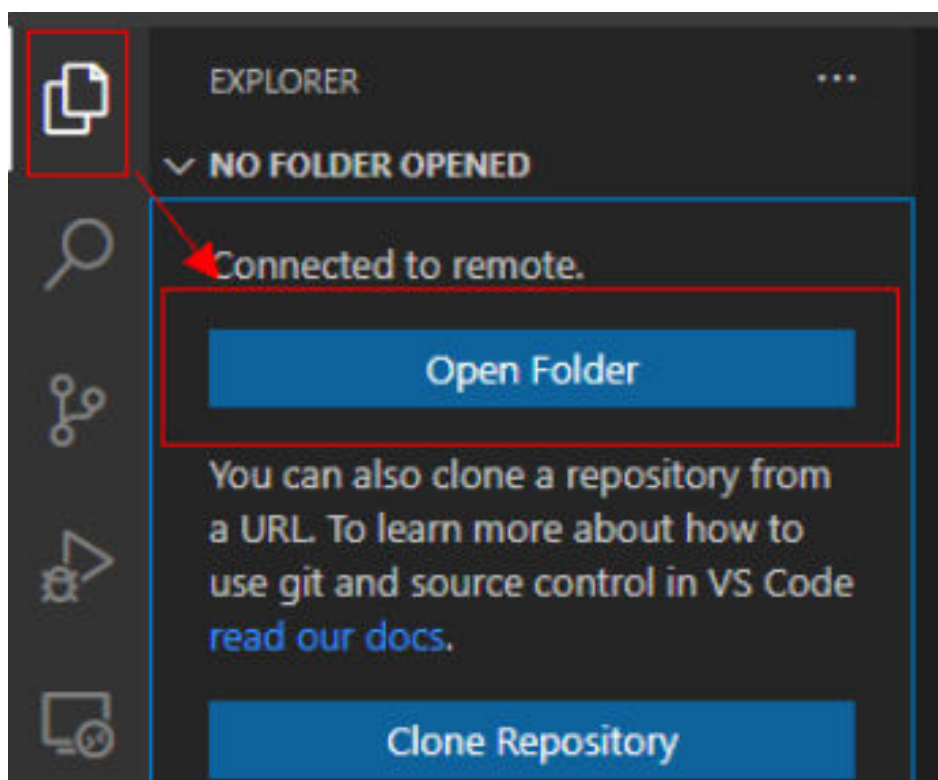
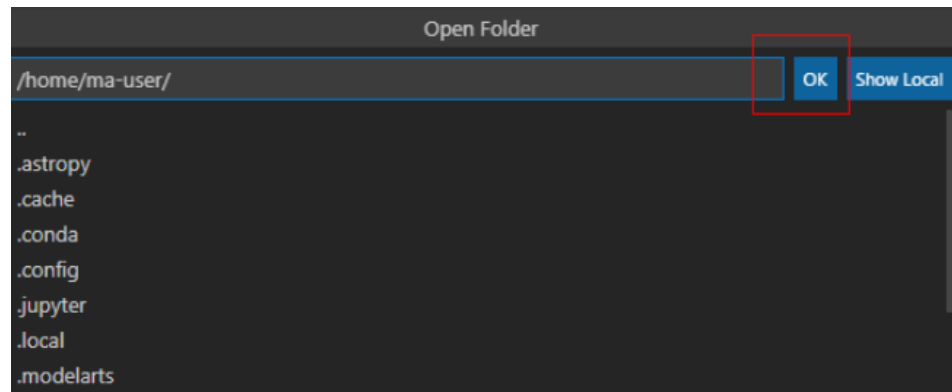
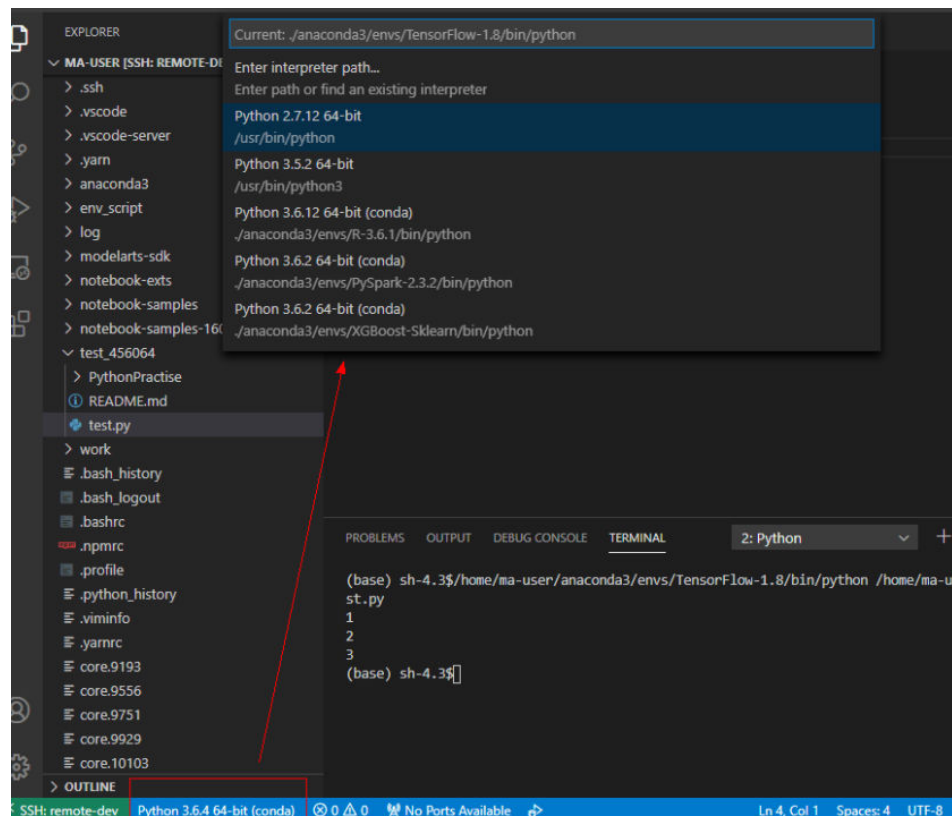


Figure 5-142 Selecting a file path

- b. In the displayed directory structure on the left of the IDE, drag the code and files you want to upload to the corresponding folders. Then, the code is uploaded to the cloud development environment.
- c. Open the code file to be debugged in VS Code. Before running the code, click the default Python version in the lower left part and select a version as required.

Figure 5-143 Selecting a Python version

2. Click the execution button to run the code. The code output is shown in the **TERMINAL** tab.
 - If a training job takes a long time to execute, run the job at the backend through the nohup command. This prevents the disconnection of an SSH

session or a network failure from affecting job execution. The following shows an example nohup command:

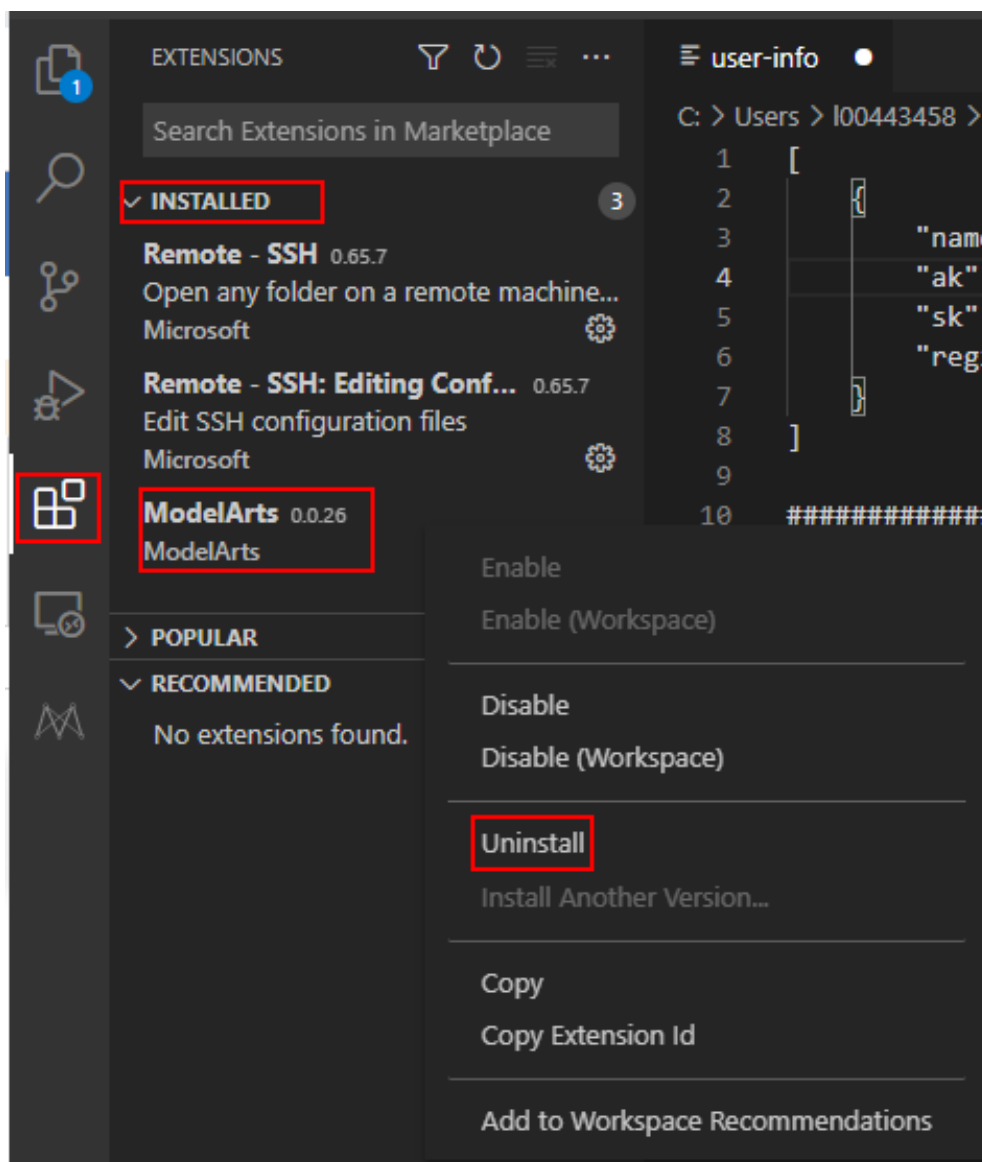
```
nohup your_train_job.sh > output.log 2>&1 & tail -f output.log
```

- To debug the code, perform the following operations:
 - i. Choose **Run > Run and Debug** on the left.
 - ii. Select the default Python code file.
 - iii. Click on the left of the code to set breakpoints.
 - iv. Debug the code according to the debug procedure which is displayed above the code, and the debug information is displayed on the left of the page.

Related Operations

For details about uninstalling the VS Code plug-in, see [Figure 5-144](#).

Figure 5-144 Uninstalling the VS Code plug-in



5.7.3 Manually Connecting to a Notebook Instance Through VS Code

A local IDE supports PyCharm and VS Code. You can use PyCharm or VS Code to remotely connect the local IDE to the target notebook instance on ModelArts for running and debugging code.

This section describes how to use VS Code to access a notebook instance.

Prerequisites

- You have downloaded and installed VS Code. For details, see [Connecting to a Notebook Instance Through VS Code](#).
- Python has been installed on your local PC or server. For details, see [VS Code official documentation](#).
- A notebook instance has been created with remote SSH enabled. Ensure that the instance is running. For details, see [Creating a Notebook Instance \(Default Page\)](#).
- The address and port number of the development environment are available. To obtain the information, go to the notebook instance details page.

Figure 5-145 Instance details page



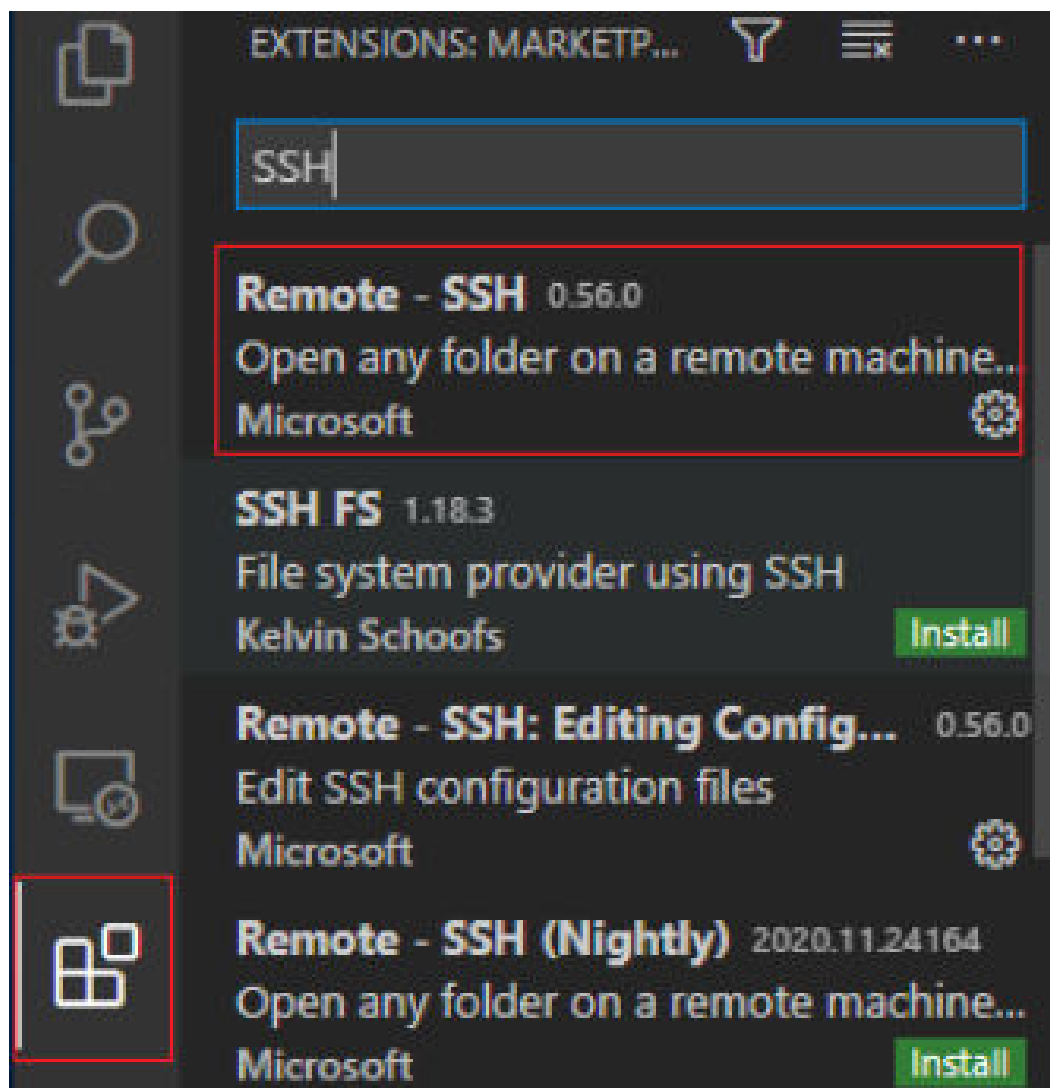
Address ssh://ma-user@dev-modelarts- 20581
Authentication KeyPair-e744

- The key pair is available.
A key pair is automatically downloaded after you create it. Securely store your key pair. If an existing key pair is lost, create a new one.

Step 1 Add the Remote-SSH Plug-in

In the local VS Code development environment, click , enter **SSH** in the search box, and click **install** of the Remote-SSH plug-in to install the plug-in.

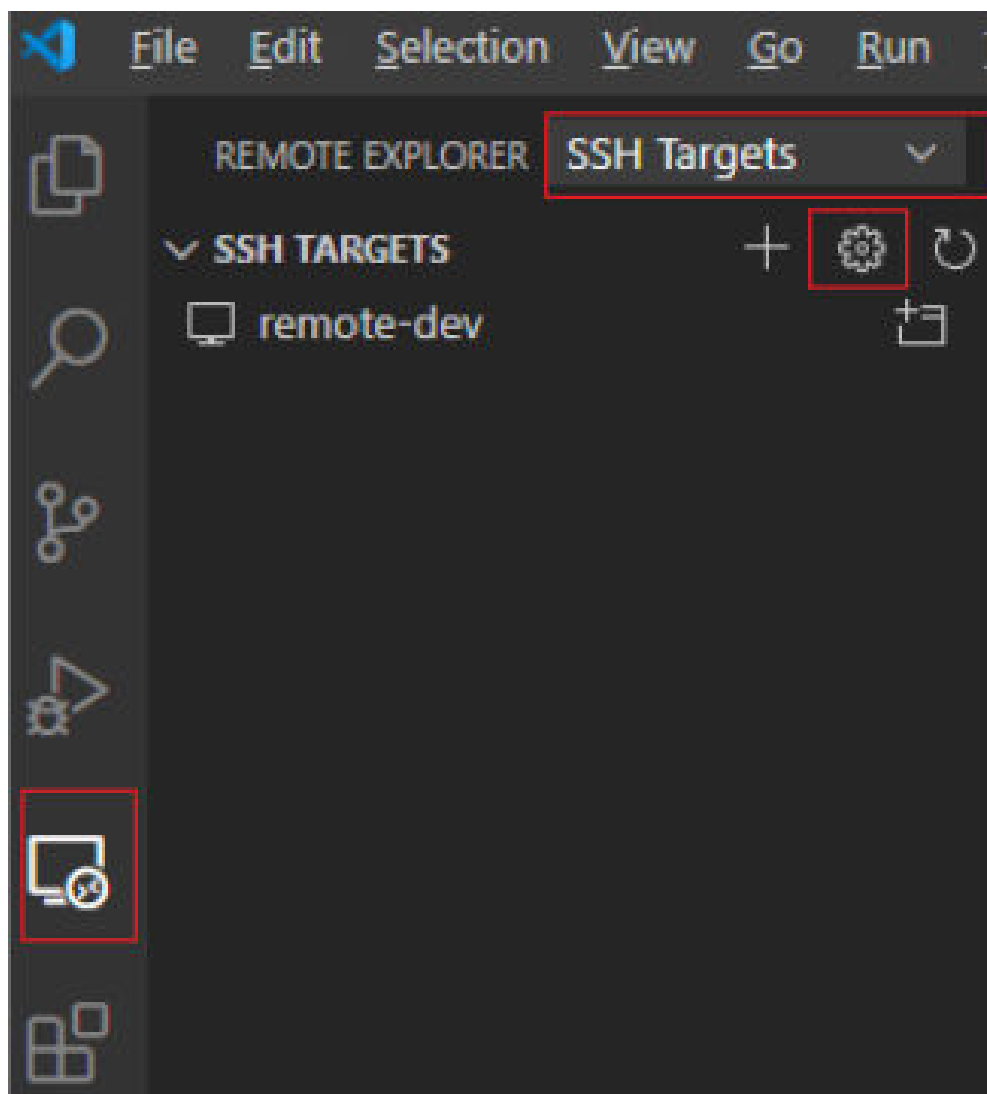
Figure 5-146 Adding the Remote-SSH plug-in



Step 2 Configure SSH

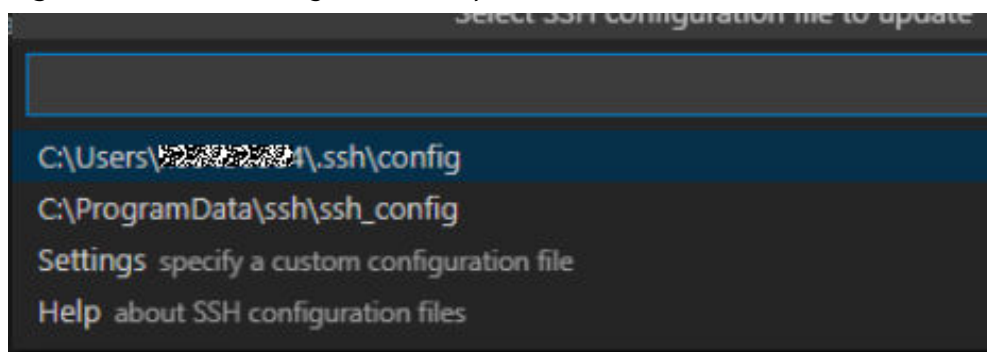
1. In the local VS Code development environment, click  on the left, select **SSH Targets** from the drop-down list box, and click . The SSH configuration file path is displayed.

Figure 5-147 Configuring SSH Targets



2. Click the SSH configuration path and configure SSH.

Figure 5-148 SSH configuration file path

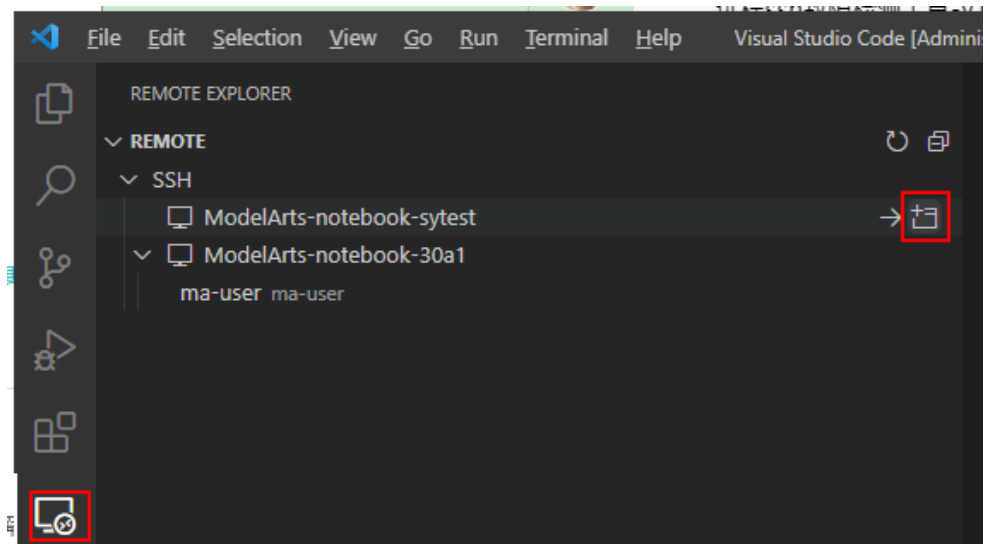


```
HOST remote-dev
  hostname <Instance connection host>
  port <Instance connection port>
  user ma-user
  IdentityFile ~/.ssh/test.pem
```

```
UserKnownHostsFile=/dev/null  
StrictHostKeyChecking no
```

- **HOST:** name of the cloud development environment
 - **HostName:** address for accessing the cloud development environment. Obtain the address on the page providing detailed information of the target notebook instance.
 - **Port:** port number for accessing the cloud development environment. Obtain the port number on the page providing detailed information of the target notebook instance.
 - **user:** **ma-user**
 - **IdentityFile:** locally stored private key file of the cloud development environment. It is the key pair file in [Prerequisites](#).
3. Return to the **SSH Targets** page, locate the target remote development environment, and click  on the right to connect to host in a new window.

Figure 5-149 Opening the development environment



On the displayed page, click **Linux**.

Figure 5-150 Selecting a notebook running environment

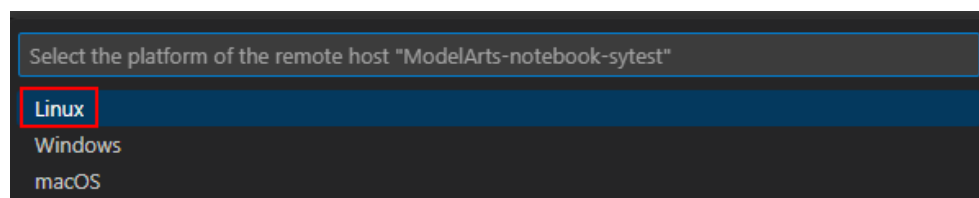
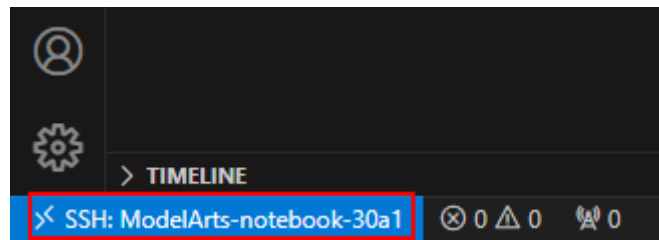
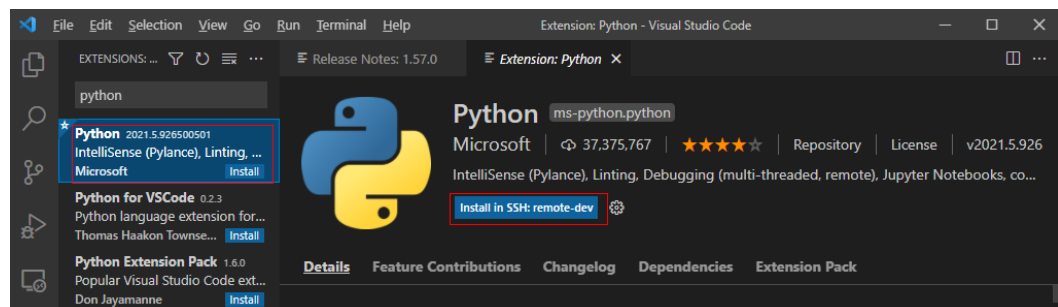


Figure 5-151 Remote connection succeeded

Step 3 Install the Python Plug-in in the Cloud Development Environment

On the displayed VS Code page, click  on the left, enter **Python** in the search box, and click **Install**.

Figure 5-152 Installing the Python plug-in in the cloud development environment

If the Python plug-in fails to be installed on the cloud, install it using an offline package.

Step 4 Install the Dependent Library for the Cloud Environment

After accessing the container environment, you can use different virtual environments, such as TensorFlow and PyTorch. However, in actual development, you need to install dependency packages. Then, you can access the environment through the terminal to perform operations.

1. In VS Code, press **Ctrl+Shift+P**.
2. Search for **Python: Select Interpreter** and select the target Python.
3. Choose **Terminal > New Terminal**. The CLI of the remote container is displayed.
4. Run the following command to install the dependency package:

```
pip install spacy
```

5.7.4 Uploading and Downloading Files in VS Code

Uploading Data to a Notebook Instance Using VS Code

If the data is less than or equal to 500 MB, directly copy the data to the local IDE.

If the data is larger than 500 MB, upload it to OBS and then to the notebook instance.

Procedure

1. Upload data to OBS. For details, see [Uploading an Object](#).

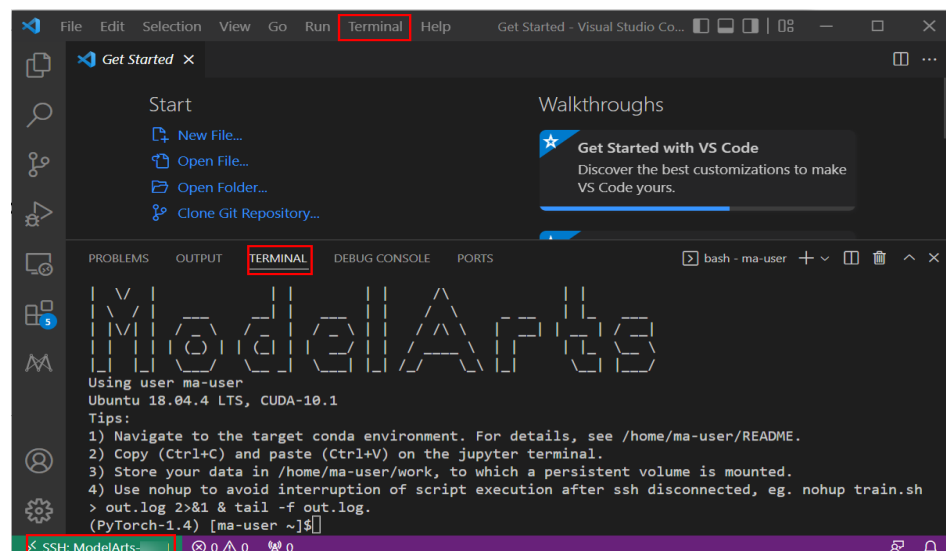
Alternatively, use ModelArts SDK in the terminal of local VS Code to upload data to OBS. To do so, click **Terminal** on the top menu bar. Enter **python** and press **Enter** in terminal to access the Python environment.

```
python
```

In the terminal of the local VS Code, use ModelArts SDK to upload the target local file to OBS. For details, [Transferring Files](#).

2. Upload the OBS file to the notebook instance. Use ModelArts SDK in the terminal of the remote VS Code environment to upload the file from OBS to a notebook instance.

Figure 5-153 Opening the terminal in the remote VS Code environment

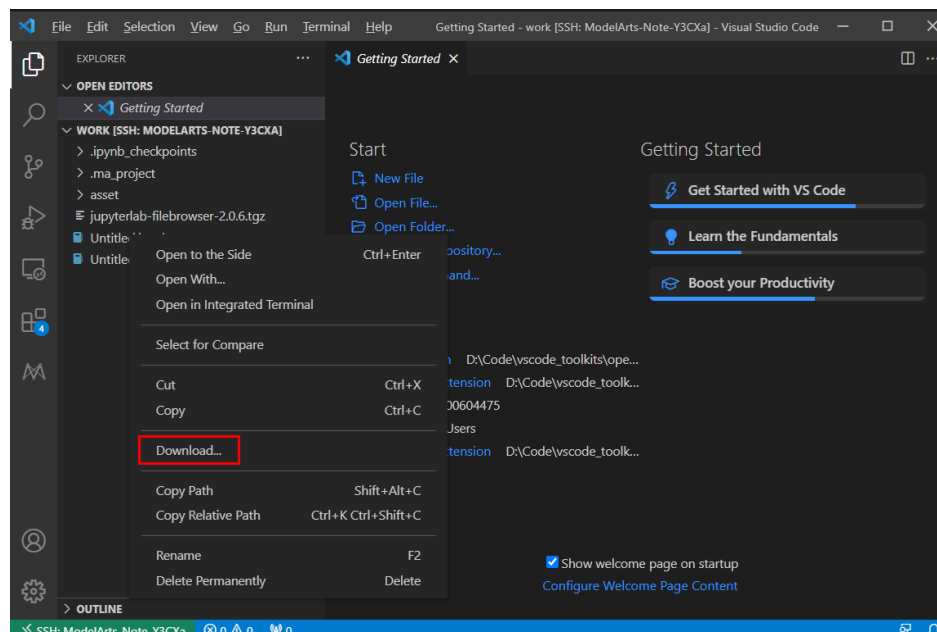


```
# Manually access the development environment using the source command.
cat /home/ma-user/README
# Select the target environment.
source /home/ma-user/miniconda3/bin/activate MindSpore-python3.7-aarch64
# Enter python and press Enter to access the Python environment.
python
```

Then, perform OBS transfer operations by referring to [Uploading a File to OBS](#).

Downloading Files from a Notebook Instance to a Local Directory

Files created in Notebook can be downloaded to a local path. In the **Project** directory of the local IDE, right-click the **Notebook2.0** project and choose **Download** from the shortcut menu to download the project file to the local PC.

Figure 5-154 Downloading files from a notebook instance to a local directory in VS Code

5.8 Using a Notebook Instance Remotely with SSH

This section describes how to use PuTTY to remotely log in to a notebook instance on the cloud in the Windows environment.

Prerequisites

- You have created a notebook instance with remote SSH enabled and whitelist configured. Ensure that the instance is running. For details, see [Creating a Notebook Instance \(Default Page\)](#).
- The address and port number of the development environment are available. To obtain this information, go to the notebook instance details page.

Figure 5-155 Instance details page

Address	ssh://ma-user@dev-modelarts- .com 32651
Authentication	KeyPair-9a64 Access address of the development environment Port number

- The key pair is available.
A key pair is automatically downloaded after you create it. Securely store your key pair. If an existing key pair is lost, create a new one.

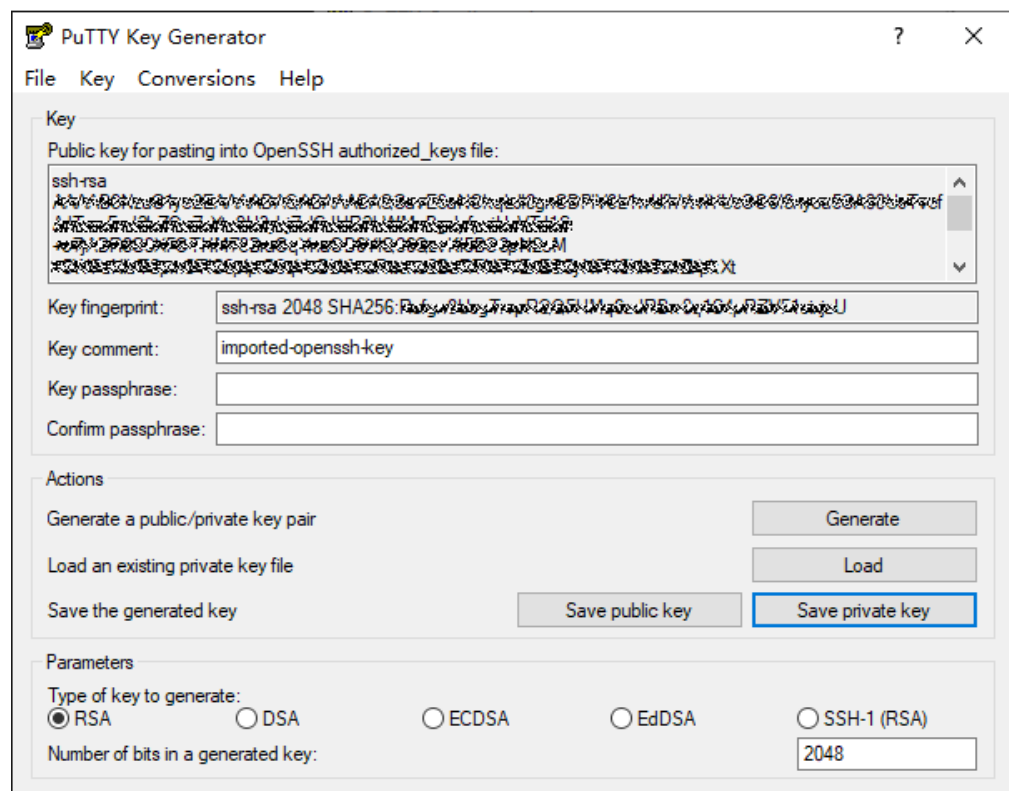
Step 1 Install the SSH Tool

[Download](#) and install the SSH remote access tool, for example, PuTTY.

Step 2 Use PuTTYgen to Convert the .pem Key Pair File to a .ppk Key Pair File

1. [Download PuTTYgen](#) and double-click it to run it.
2. Click **Load** to load the .pem key file created and saved during notebook instance creation.
3. Click **Save private key** to save the generated .ppk file. The file name can be customized, for example, **key.ppk**.

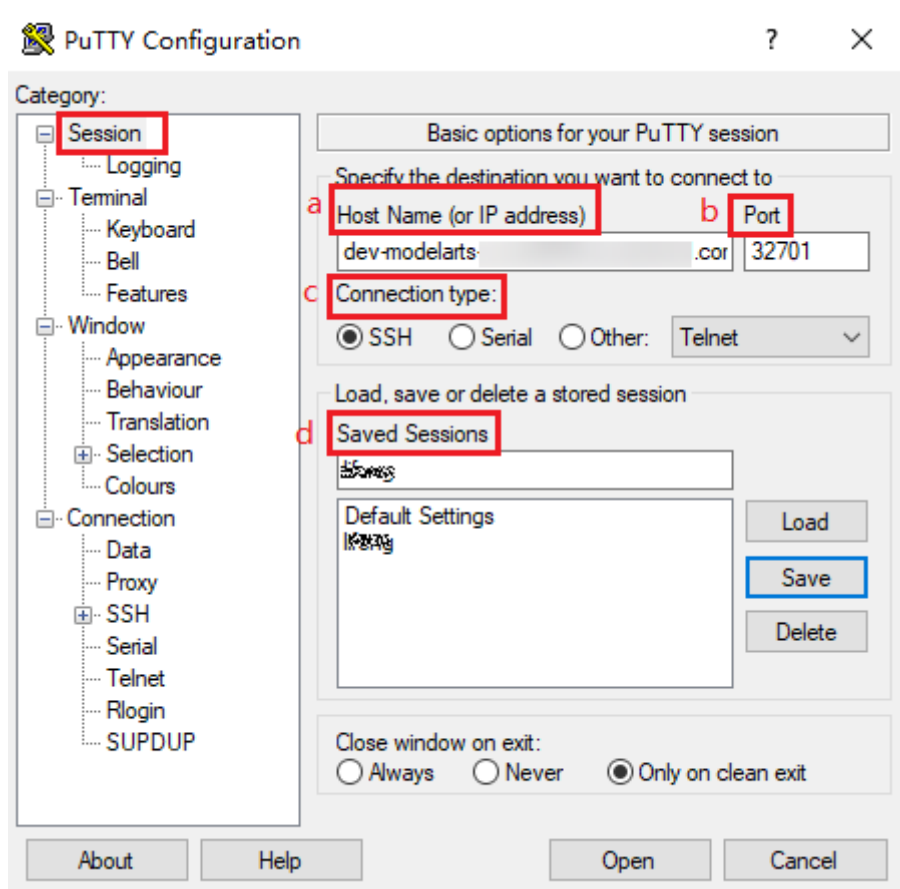
Figure 5-156 Converting the .pem key pair file to a .ppk key pair file



Step 3 Use SSH to Connect to a Notebook Instance

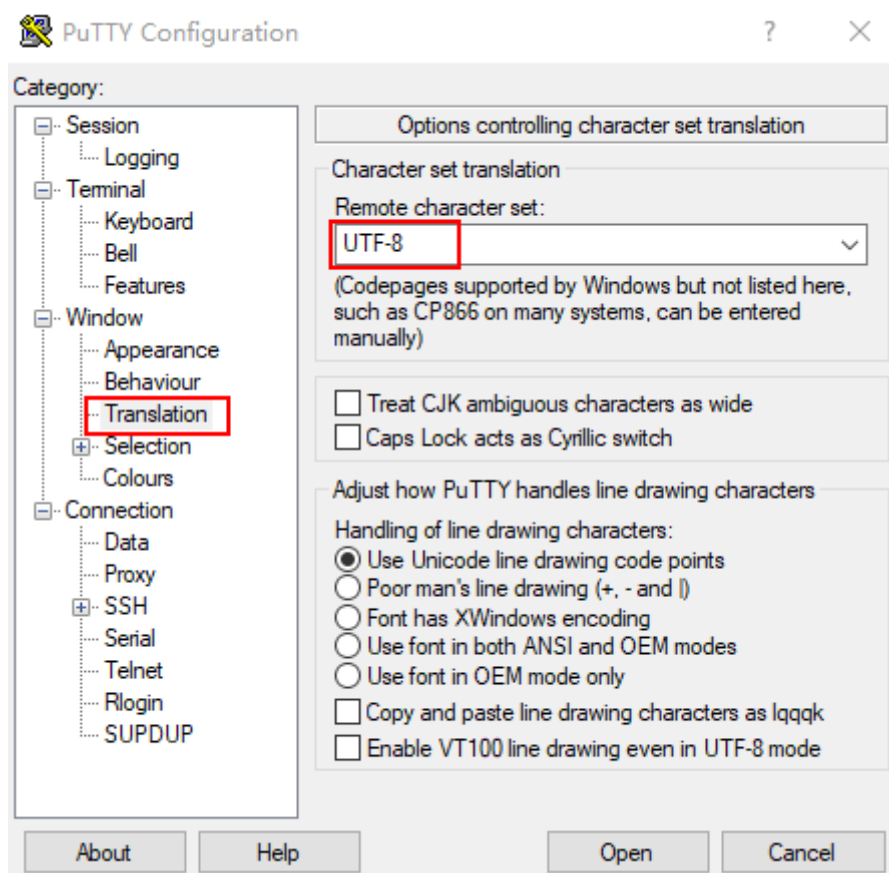
1. Run PuTTY.
2. Click **Session** and set the following parameters:
 - a. **Host Name (or IP address)**: address for accessing the in-cloud notebook instance. Obtain the address on the page providing detailed information of the target notebook instance .
 - b. **Port**: port number for accessing the in-cloud notebook instance. Obtain the port number on the page providing detailed information of the target notebook instance, for example, **32701**.
 - c. **Connection type**: Choose **SSH**.
 - d. **Saved Sessions**: task name, which can be clicked for remote access when you use PuTTY next time

Figure 5-157 Configuring Session



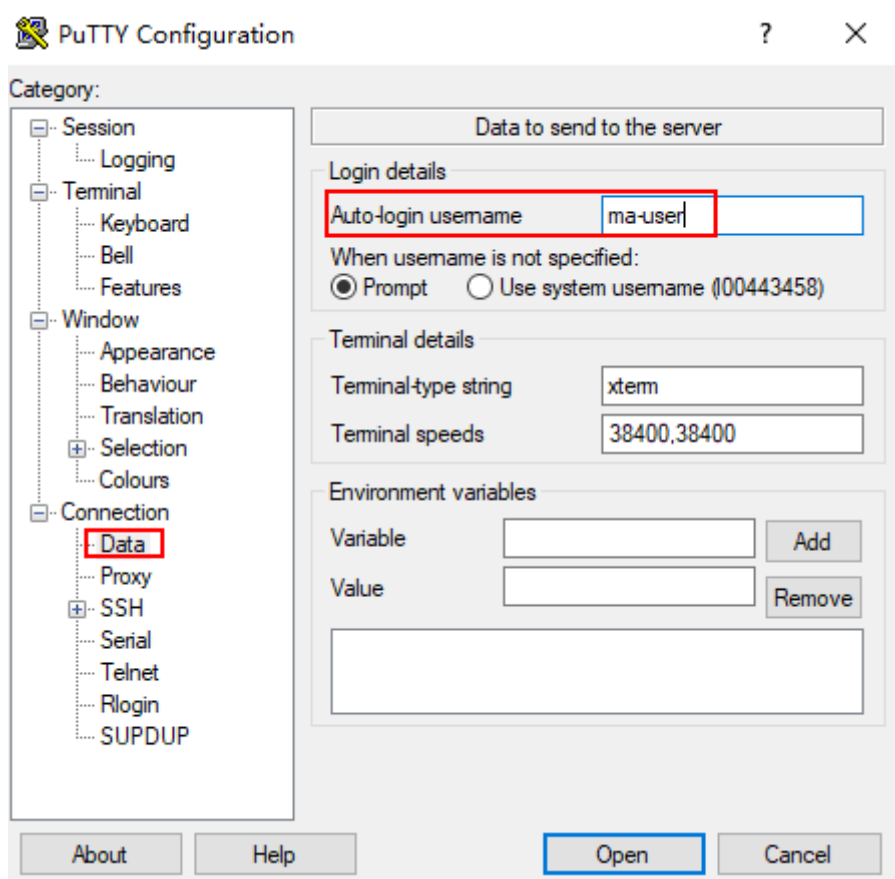
3. Choose **Window > Translation** and select **UTF-8** from the drop-down list box in the **Remote character set** area.

Figure 5-158 Setting the character format

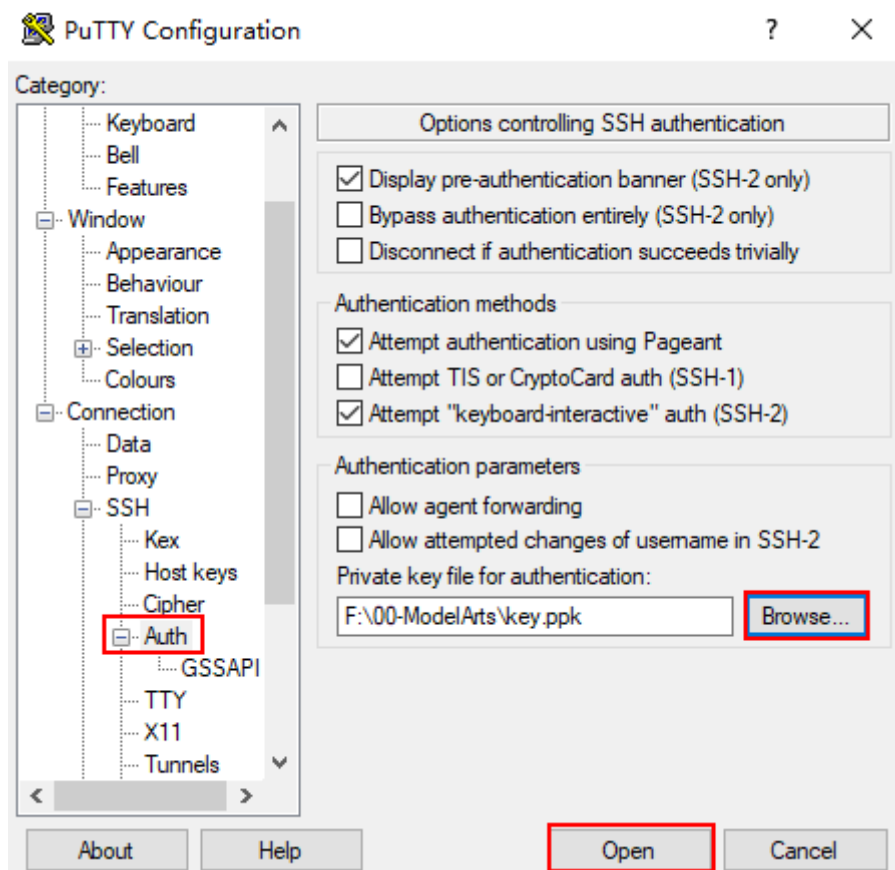


4. Choose **Connection** > **Data** and enter **ma-user** for **Auto-login username**.

Figure 5-159 Entering a username

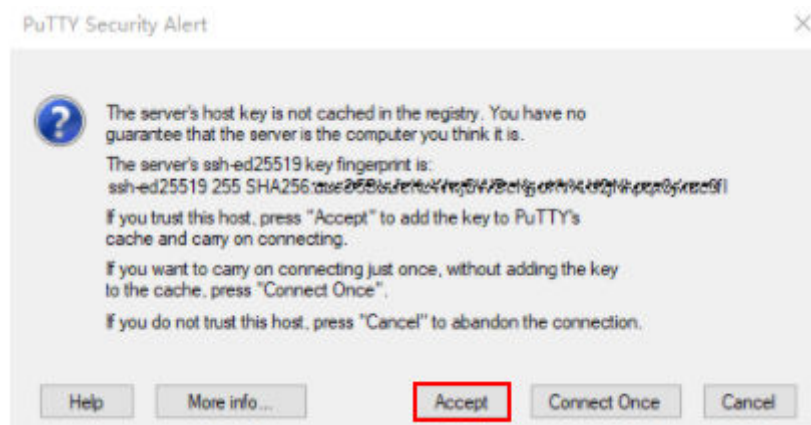


5. Choose **Connection** > **SSH** > **Auth**, click **Browse**, and select the .ppk file generated in [step 2](#).



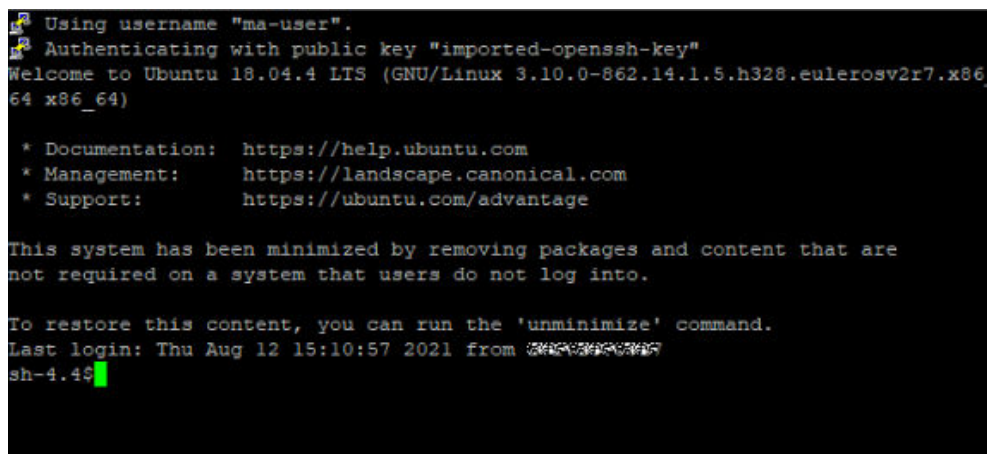
- Click **Open**. If you are logging in to the instance for the first time, PuTTY displays a security warning dialog box, asking if you want to accept the instance security certificate. Click **Accept** to save the certificate to your local registry.

Figure 5-160 Asking if you want to accept the instance security certificate



7. Connect to the notebook instance.

Figure 5-161 Connecting to a notebook instance

A terminal window showing an SSH connection. The output includes: 'Using username "ma-user".', 'Authenticating with public key "imported-openssh-key"', 'Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 3.10.0-862.14.1.el7.x86_64 x86_64)', documentation and support links, a message about system minimization, and the 'unminimize' command. The prompt is 'sh-4.4\$' with a green cursor.

```
Using username "ma-user".
Authenticating with public key "imported-openssh-key"
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 3.10.0-862.14.1.el7.x86_64 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Thu Aug 12 15:10:57 2021 from 192.168.1.100
sh-4.4$
```

5.9 ModelArts CLI Command Reference

5.9.1 ModelArts CLI Commands

Description

ModelArts CLI, also called ma-cli, is a cross-platform command line tool used to connect to ModelArts and run management commands on ModelArts resources. You can use the interactive command prompt or script to run commands on a terminal. ma-cli allows you to interact with cloud services through ModelArts notebook and on-premises VMs. You can run ma-cli commands for command autocomplete and authentication, as well as creating images, submitting ModelArts training jobs and DLI Spark jobs, and copying OBS data.

Application Scenarios

- ma-cli has been integrated into ModelArts notebook and can be directly used. Log in to the ModelArts console, choose **Development Workspace > Notebook**, create a notebook instance, start a terminal, and run ma-cli commands.
- In local Windows or Linux, install ma-cli and then use it on a local terminal. For details, see [\(Optional\) Installing ma-cli Locally](#).

NOTE

- ma-cli cannot be used in Git Bash.
- Terminals such as Linux Bash, Zsh, Fish, WSL, and PowerShell are recommended. To ensure the security of your sensitive information, it is important to prevent any potential leakage when using terminals.

Command Preview

```
$ ma-cli -h
Usage: ma-cli [OPTIONS] COMMAND [ARGS]...

Options:
  -V, --version  1.2.1
```

```
-C, --config-file TEXT  Configure a file path for authorization.
-D, --debug             Debugging mode, in which the full stack trace will be displayed when an error occurs.
-P, --profile TEXT      CLI connection profile to be used. The default profile is DEFAULT.
-h, -H, --help          Show the help information and exit.

Commands:
configure               Configure authentication and endpoints for the CLI.
image                   Obtain registered images, register or unregister images, debug images, and create images in
                        Notebook.
obs-copy                Copy files or directories between OBS and a local path.
ma-job                  Submit ModelArts jobs and obtain job details.
dli-job                 Submit DLI spark jobs and obtain job details.
auto-completion         Auto complete ma-cli command in terminal, support "bash(default)/zsh/fish".
```

Among the preceding parameters, parameters **-C**, **-D**, **-P**, and **-h** are globally optional.

- **-C** indicates that you can manually specify the authentication configuration file when running this command. By default, the **~/.modelarts/ma-cli-profile.yaml** configuration file is used.
- **-P** indicates a group of authentication information in the authentication file. The default value is **DEFAULT**.
- **-D** indicates whether to enable the debugging mode (disabled by default). After the debugging mode is enabled, the error stack information of the command will be printed. If this mode is disabled, only the error information will be printed.
- **-h** indicates that the help information about the command will be displayed.

Commands

Table 5-24 ma-cli commands

Command	Description
configure	ma-cli authentication using a username and password or an SK/SK
image	ModelArts image creation, registration, and registered image query
obs-copy	Copying files or folders between a local path and OBS
ma-job	Managing ModelArts training jobs, including job submission and resource query
dli-job	DLI Spark job submission and resource management
auto-completion	Command autocomplete

5.9.2 (Optional) Installing ma-cli Locally

Application Scenarios

This document describes how to install ma-cli on Windows.

Step 1: Install ModelArts SDKs

Install ModelArts SDKs by referring to [Installing the ModelArts SDK Locally](#).

Step 2: Download ma-cli

1. [Download the ma-cli software package](#).
2. Verify the software package signature.
 - a. [Download the signature verification file of the software package](#).
 - b. Install OpenSSL and run the following command to verify the signature:

```
openssl cms -verify -binary -in D:\ma_cli-latest-py3-none-any.whl.cms -inform DER -content D:\ma_cli-latest-py3-none-any.whl -noverify > ./test
```

NOTE

In this example, the software package is stored in **D:**. Replace it with the actual path.

```
st $openssl cms -verify -binary -in package.tar.gz.cms -signer "root" -inform DER -content package.tar.gz -noverify > ./test
CMS Verification successful
```

Step 3: Install ma-cli

1. Run **python --version** in the command prompt of your local environment to check whether Python has been installed. The Python version must be later than 3.7.x and earlier than 3.10.x. Version 3.7.x is recommended.

```
C:\Users\xxx>python --version
Python 3.7.4
```
2. Run **pip --version** to check whether the general package management tool pip is available.

```
C:\Users\xxx>pip --version
pip 20.0.2 from c:\users\xxx\appdata\local\programs\python\python37\lib\site-packages\pip (python 3.7.4)
```
3. Install ma-cli.

pip install {Path to the ma-cli software package}\ma_cli-latest-py3-none-any.whl

```
C:\Users\xxx>pip install C:\Users\xxx\Downloads\ma_cli-latest-py3-none-any.whl
.....
Successfully installed ma_cli-1.0.0
```

When ma-cli is installed, dependency packages are installed by default. If message "Successfully installed" is displayed, ma-cli has been installed.

NOTE

If an error message is displayed during the installation, indicating that a dependency package is missing, run the following command to install the dependency package as prompted:

pip install xxxx

xxxx is the name of the dependency package.

5.9.3 Autocompletion for ma-cli Commands

CLI autocomplete enables you to get a list of supported **ma-cli** commands by typing a command prefix and pressing **Tab** on your terminal. Autocomplete for **ma-cli** commands needs to be enabled in Terminal. After running the **ma-cli auto-completion** command, you can copy and run the commands as prompted

on the current terminal to automatically complete the **ma-cli** commands. Bash, Fish, and Zsh shells are supported. The default shell is Bash.

Take the Bash command as an example. Run the **eval "\$(_MA_CLI_COMPLETE=bash_source ma-cli)"** command in Terminal to enable autocomplete.

```
eval "$(_MA_CLI_COMPLETE=bash_source ma-cli)"
```

Run the **ma-cli auto-completion Zsh** or **ma-cli auto-completion Fish** command to view the autocomplete command in Zsh or Fish.

Available Commands

```
$ ma-cli auto-completion -h
Usage: ma-cli auto-completion [OPTIONS] [[Bash|Zsh|Fish]]
```

Auto complete ma-cli command in terminal.

Example:

```
# print bash auto complete command to terminal
ma-cli auto-completion Bash
```

Options:

```
-H, -h, --help Show this message and exit.
```

```
# By default, the autocomplete command for Bash is displayed.
```

```
$ ma-cli auto-completion
```

Tips: please paste following shell command to your terminal to activate auto completion.

```
[ OK ] eval "$(_MA_CLI_COMPLETE=bash_source ma-cli)"
```

```
# After the preceding command is executed, autocomplete has been enabled on the terminal.
```

```
$ eval "$(_MA_CLI_COMPLETE=bash_source ma-cli)"
```

```
# The autocomplete command for Fish is displayed.
```

```
$ ma-cli auto-completion Fish
```

Tips: please paste following shell command to your terminal to activate auto completion.

```
[ OK ] eval (env _MA_CLI_COMPLETE=fish_source ma-cli)
```

5.9.4 ma-cli Authentication

Overview

- VMs and personal computers require the configuration of authentication. Both a username and password (default) and an AK/SK can be used for authentication.
- When using an account for authentication, specify a username and password. When using an IAM account for authentication, specify an account, username, and password.
- In ModelArts notebook, you do not need to manually configure authentication because an agency is used for authentication by default.
- If you have configured authentication in ModelArts notebook, the specified authentication is preferentially used.

 NOTE

To ensure the security of your sensitive information, it is important to prevent any potential leakage during authentication.

CLI Parameters

```
$ ma-cli configure -h
Usage: ma-cli configure [OPTIONS]

Options:
  -auth, --auth [PWD|AKSK|ROMA]  Authentication type.
  -rp, --region-profile PATH      ModelArts region file path.
  -a, --account TEXT              Account of an IAM user.
  -u, --username TEXT             Username of an IAM user.
  -p, --password TEXT             Password of an IAM user
  -ak, --access-key TEXT          User access key.
  -sk, --secret-key TEXT          User secret key.
  -r, --region TEXT               The region you want to visit.
  -pi, --project-id TEXT          User project id.
  -C, --config-file TEXT          Configure file path for authorization.
  -D, --debug                     Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT              CLI connection profile to use. The default profile is "DEFAULT".
  -h, -H, --help                  Show this message and exit.
```

Table 5-25 Authentication CLI parameters

Parameter	Type	Man dator y	Description
-auth / --auth	String	No	Authentication mode, which can be PWD (username and password) or AKSK (AK/SK). The default value is PWD .
-rp / --region-profile	String	No	ModelArts region configuration file
-a / --account	String	No	IAM tenant account, which needs to be specified when authentication using an IAM account is used. It is required in authentication using a username and password.
-u / --username	String	No	Username, which is a username or an IAM username for authentication using an account or an IAM account. It is required in authentication using a username and password.
-p / --password	String	No	Password, which is required in authentication using a username and password
-ak / --access-key	String	No	Access key, which is required in authentication using an AK/SK
-sk / --secret-key	String	No	Secret key, which is required in authentication using an AK/SK

Parameter	Type	Mandatory	Description
-r / --region	String	No	Region name. If this parameter is left blank, the value of the REGION_NAME environment variable will be used by default.
-pi / --project-id	String	No	Project ID. If this parameter is left blank, the region value (default) or the value of the PROJECT_ID environment variable will be used.
-P / --profile	String	No	Authentication configuration, which defaults to DEFAULT
-C / --config-file	String	No	Local path to the configuration file, which defaults to ~/.modelarts/ma-cli-profile.yaml

Authentication Using Username and Password

The following describes how to use the **ma-cli configure** command on a VM to configure authentication using the user name and password.

NOTE

In the following example, any string with **\${}** is a variable. You can specify a value.

For example, **\${your_password}** indicates that you need to type your password.

```
# The DEFAULT authentication configuration is used by default. You need to type the account, username,
and password one by one. If the account and username are not required, press Enter to skip them.
$ ma-cli configure --auth PWD --region ${your_region}
account: ${your_account}
username: ${your_username}
password: ${your_password} # The input is not displayed on the console.
```

Authentication Using an AK/SK

This command uses an AK/SK for authentication, which means you have to enter them interactively. Your AK/SK will not be visible on the console.

CAUTION

In the following example, any string with **\${}** is a variable. You can specify a value. For example, you need to replace **\${access key}** with your access key.

```
ma-cli configure --auth AKSK
access key [***]: ${access key}
secret key [***]: ${secret key}
```

After the authentication command is executed, the authentication information will be saved in the **~/.modelarts/ma-cli-profile.yaml** configuration file.

5.9.5 ma-cli image Commands for Building Images

The **ma-cli image** command can be used to obtain registered images, obtain or load image creation templates, create images using Dockerfiles, obtain or clear image creation caches, register or deregister images, and debug whether images can be used in notebook instances. For details, run the **ma-cli image -h** command.

Commands for Creating an Image

```
$ ma-cli image -h
Usage: ma-cli image [OPTIONS] COMMAND [ARGS]...
    Obtain registered images, register or unregister images, debug images, and create images in Notebook.

Options:
  -H, -h, --help  Show this message and exit.

Commands:
  add-template, at  List build-in dockerfile templates.
  build            Build docker image in Notebook.
  debug           Debug SWR image as a Notebook in ECS.
  df              Query disk usage.
  get-image, gi   Query registered image in ModelArts.
  get-template, gt List build-in dockerfile templates.
  prune          Prune image build cache.
  register        Register image to ModelArts.
  unregister      Unregister image from ModelArts.
```

Table 5-26 Commands for creating an image

Command	Description
get-template	Obtain an image creation template.
add-template	Load an image creation template.
get-image	Obtain registered ModelArts images.
register	Register SWR images with ModelArts image management.
unregister	Deregister a registered image from ModelArts image management.
build	Build an image using a Dockerfile (only supported in ModelArts Notebook).
df	Obtain image creation cache, which can only be used in ModelArts notebook.
prune	Clear image creation cache, which can only be used in ModelArts notebook.

Command	Description
debug	Debug an SWR image on an ECS to check whether the image can be used in ModelArts notebook. (Only the ECSs with Docker installed can be used.)

Using ma-cli image get-template to Query an Image Building Template

ma-cli provides some common image building templates, which contain the guidance for developing Dockerfiles on ModelArts notebook.

```
$ ma-cli image get-template -h
Usage: ma-cli image get-template [OPTIONS]

List build-in dockerfile templates.

Example:

# List build-in dockerfile templates
ma-cli image get-template [--filter <filter_info>] [--page-num <yourPageNum>] [--page-size <yourPageSize>]

Options:
  --filter TEXT          filter by keyword.
  -pn, --page-num INTEGER RANGE Specify which page to query. [x>=1]
  -ps, --page-size INTEGER RANGE The maximum number of results for this query. [x>=1]
  -D, --debug            Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT     CLI connection profile to use. The default profile is "DEFAULT".
  -H, -h, --help        Show this message and exit.
(PyTorch-1.4) [ma-user work]$
```

Table 5-27 Parameters

Parameter	Data Type	Mandatory	Description
--filter	String	No	Template name keyword for filtering templates.
-pn / --page-num	Int	No	Image page index. The default value is page 1.
-ps / --page-size	Int	No	Number of images displayed on each page. The default value is 20 .

Example: Obtain an image building template.

ma-cli image get-template

```
(PyTorch-1.8) [ma-user work]$ma-cli image get-template
Template Name      Description
-----
customize_from_ubuntu_18.04_to_modelarts  Add ma-user, apt install packages and create a new conda environment with pip based on scratch ubuntu 18.04
upgrade_current_notebook_apt_packages     Install apt packages like ffmpeg, gcc-8, g++-8 based on current Notebook image
migrate_3rd_party_image_to_modelarts       General template for migrating your own or open source image to ModelArts
migrate_official_torch_110_cu113_image_to_modelarts  Reconstructing and migrating the official torch 1.10.0 with cud11.3 image to ModelArts
build_handwritten_number_inference_application  Create a new AI application, used to generate an image to deploy and infer in ModelArts
update_dli_image_pip_package               Install pip packages based on DLI image
forward_compat_cuda_11_image_to_modelarts  Migrate and forward compat cuda-11.x to ModelArts by upgrading only user-mode CUDA components
```

Using ma-cli image add-template to Load an Image Building Template

The **add-template** command is used to load image templates to a specified folder. By default, the path where the current command is located is used,

for example, `${current_dir}/.ma/${template_name}/`. You can also use **--dest** to specify the path. If a template folder with the same name already exists in the target path, use the **--force** | **-f** parameter to forcibly overwrite the existing template folder.

```
$ ma-cli image add-template -h
Usage: ma-cli image add-template [OPTIONS] TEMPLATE_NAME

Add buildin dockerfile templates into disk.

Example:

# List build-in dockerfile templates
ma-cli image add-template customize_from_ubuntu_18.04_to_modelarts --force

Options:
  --dst TEXT      target save path.
  -f, --force      Override templates that has been installed.
  -D, --debug      Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT CLI connection profile to use. The default profile is "DEFAULT".
  -h, -H, --help   Show this message and exit.
```

Table 5-28 Parameters

Parameter	Data Type	Mandatory	Description
--dst	String	No	Target path for loading a template. The current path is used by default.
-f / --force	Bool	No	Whether to forcibly overwrite an existing template with the same name. By default, the template is not overwritten.

Example: Load the **customize_from_ubuntu_18.04_to_modelarts** image building template.

```
ma-cli image add-template customize_from_ubuntu_18.04_to_modelarts
```

```
(PyTorch-1.8) [ma-user work]$ma-cli image add-template customize_from_ubuntu_18.04_to_modelarts
[ OK ] Successfully add configuration template [ customize_from_ubuntu_18.04_to_modelarts ] under folder [ /home/ma-user/work/.ma/customize_from_ubuntu_18.04_to_modelarts ]
```

Using ma-cli image get-image to Query Registered ModelArts Images

A path to a base image is provided in a Dockerfile typically. Public images and SWR public or private images can be obtained from open-source image repositories such as Docker Hub. ma-cli allows you to obtain ModelArts preset images, registered images, and their SWR addresses.

```
$ma-cli image get-image -h
Usage: ma-cli image get-image [OPTIONS]

Get registered image list.
```

Example:

```
# Query images by image type and only image id, show name and swr_path  
ma-cli image get-image --type=DEDICATED
```

```
# Query images by image id  
ma-cli image get-image --image-id ${image_id}
```

```
# Query images by image type and show more information  
ma-cli image get-image --type=DEDICATED -v
```

```
# Query images by image name  
ma-cli image get-image --filter=torch
```

Options:

```
-t, --type [BUILD_IN|DEDICATED|ALL]      Image type(default ALL)  
-f, --filter TEXT                        Image name to filter  
-v, --verbose                            Show detailed information on image.  
-i, --image-id TEXT                      Get image details by image id  
-n, --image-name TEXT                    Get image details by image name  
-wi, --workspace-id TEXT                 The workspace where you want to query image(default "0")  
-pn, --page-num INTEGER RANGE            Specify which page to query [x>=1]  
-ps, --page-size INTEGER RANGE           The maximum number of results for this query [x>=1]  
-C, --config-file PATH                   Configure file path for authorization.  
-D, --debug                             Debug Mode. Shows full stack trace when error occurs.  
-P, --profile TEXT                       CLI connection profile to use. The default profile is "DEFAULT".  
-H, -h, --help                          Show this message and exit.
```

Table 5-29 Parameters

Parameter	Data Type	Man datory	Description
-t / --type	String	No	Type of the images to be obtained. The options are BUILD_IN , DEDICATED , and ALL . • BUILD_IN: preset images • DEDICATED: custom images registered with ModelArts • ALL: all images
-f / --filter	String	No	Image name keyword for filtering images.
-v / --verbose	Bool	No	Whether to display detailed information. It is disabled by default.
-i / --image-id	String	No	ID of the image whose details are to be obtained.
-n / --image-name	String	No	Name of the image whose details are to be obtained.
-wi / --workspace-id	String	No	Workspace in which the image information is to be obtained.
-pn / --page-num	Int	No	Image page index. The default value is page 1.

Parameter	Data Type	Man dato ry	Description
-ps / --page-size	Int	No	Number of images displayed on each page. The default value is 20 .

Example: Obtain custom images registered with ModelArts.

```
ma-cli image get-image --type=DEDICATED
```

```
(PyTorch-1.8) [ma-user work]$ma-cli image get-image --type=DEDICATED
```

INDEX	IMAGE ID	NAME	SWR PATH
1	c857e5a8	ifc5e3d002f 0314test	/notebook_test/0314test:1.0.0
2	193b2557	d39093a811 0328	.com/notebook_test/0328:1
3	171fe036	3b37e9aa7c 0926	i_modelarts_y00218826_05/0926:1
4	1b48bb0a	689b0a7267 0926	_modelarts_y00218826_05/0926:111
5	c8667cf0	d2e3563107 1	/ei_modelarts_y00218826_05/1:6
6	3e6cda6a	a360eea80e 1	/ei_modelarts_y00218826_05/1:1
7	42e86ca5	ec198be968 111	com/notebook_test/111:1227
8	0f349cef	c411011ef2 11111110801	otebook_test/11111110801:111111
9	3a082e32	4f485aad66 112121	modelarts_y00218826_05/112121:123
10	db0d02f6	74eb00e1ce 1203	om/notebook_test/1203:1.2.3
11	031dc02e	ld92cd457d8 1227	com/notebook_test/1227:111
12	f7d95648	7aaec8b1cc 1227	com/notebook_test/1227:888
13	2f720610	a1d1db9d7d 1227	com/notebook_test/1227:6666
14	42221bf2	i22d726d270 1229	com/notebook_test/1229:123
15	70deea1e	70b2414ae7 123	om/mindspore-dis-train/123:2
16	e6cc5414	ce318069f4 123	com/notebook_test/123:45678
17	6e7a86c9	319fb3bb28 1234	com/notebook_test/1234:666
18	ec036306	8c9dc6b391 1234	.com/notebook_test/1234:1
19	b37f8f3b	7a9941c978 441211	com/notebook_test/441211:11
20	d5acd51b	lef16534d68 aaa	com/notebook_test/aaa:1.1.1

Using ma-cli image build to Build an Image in ModelArts Notebook

Run the **ma-cli image build** command to build an image based on a specified Dockerfile. This command is available only on ModelArts notebook instances.

```
$ ma-cli image build -h
```

```
Usage: ma-cli image build [OPTIONS] FILE_PATH
```

Build docker image in Notebook.

Example:

```
# Build a image and push to SWR
```

```
ma-cli image build .ma/customize_from_ubuntu_18.04_to_modelarts/Dockerfile -swr my_organization/  
my_image:0.0.1
```

```
# Build a image and push to SWR, dockerfile context path is current dir
```

```
ma-cli image build .ma/customize_from_ubuntu_18.04_to_modelarts/Dockerfile -swr my_organization/  
my_image:0.0.1 -context .
```

```
# Build a local image and save to local path and OBS
```

```
ma-cli image build .ma/customize_from_ubuntu_18.04_to_modelarts/Dockerfile --target ./build.tar --
```

```
obs_path obs://bucket/object --swr-path my_organization/my_image:0.0.1
```

Options:

- t, --target TEXT Name and optionally a tag in the 'name:tag' format.
- swr, --swr-path TEXT SWR path without swr endpoint, eg:organization/image:tag. [required]
- context DIRECTORY build context path.
- arg, --build-arg TEXT build arg for Dockerfile.
- obs, --obs-path TEXT OBS path to save local built image.
- f, --force Force to overwrite the existing swr image with the same name and tag.
- C, --config-file PATH Configure file path for authorization.
- D, --debug Debug Mode. Shows full stack trace when error occurs.
- P, --profile TEXT CLI connection profile to use. The default profile is "DEFAULT".
- H, -h, --help Show this message and exit.

Table 5-30 Parameters

Parameter	Data Type	Mandatory	Description
FILE_PATH	String	Yes	Path where the Dockerfile is stored.
-t / --target	String	No	Local path for storing the generated TAR package. The current directory is used by default.
-swr / --swr-path	String	Yes	SWR image name, which is in the format of "organization/image_name:tag". This parameter can be omitted when a TAR package is saved for building an image.
--context	String	No	Context used for copying data when a Dockerfile is used.
-arg / --build-arg	String	No	Parameter for building an image. If there are multiple parameters, run --build-arg VERSION=18.04 --build-arg ARCH=X86_64 .
-obs / --obs-path	String	No	OBS path for automatically uploading the generated TAR package.
-f / --force	Bool	No	Whether to forcibly overwrite an existing SWR image with the same name. By default, the SWR image is not overwritten.

Example: Build an image in ModelArts notebook.

```
ma-cli image build .ma/customize_from_ubuntu_18.04_to_modelarts/Dockerfile -swr  
notebook_test/my_image:0.0.1
```

In this command, **.ma/customize_from_ubuntu_18.04_to_modelarts/Dockerfile** is the path where the Dockerfile is stored, and **notebook_test/my_image:0.0.1** is the SWR path of the new image.

```
(PyTorch-1.8) [ma-user work]$ma-cli image build .ma/customize_from_ubuntu_18.04_to_modelarts/Dockerfile -swr notebook_test/my_image:0.0.1
[*] Building 4.3s (8/8) FINISHED
-> [internal] load .dockerignore
-> => transferring context: 2B
-> [internal] load build definition from Dockerfile
-> => transferring dockerfile: 3.29kB
-> [internal] load metadata for swr.cn-north-7.myhuaweicloud.com/atelier/ubuntu:18.04
-> [auth] atelier/ubuntu:pull token for swr.cn-north-7.myhuaweicloud.com
-> [1/2] FROM swr.cn-north-7.myhuaweicloud.com/atelier/ubuntu:18.04@sha256:b58746c8a89938b8c9f5b77de3b8cf1fe78210c696ab03a1442e235eea65d84f
-> => resolve swr.cn-north-7.myhuaweicloud.com/atelier/ubuntu:18.04@sha256:b58746c8a89938b8c9f5b77de3b8cf1fe78210c696ab03a1442e235eea65d84f
-> => sha256:2910811b6c4227c2f42aaea9a3dd5f53b1d469f67e2cf7e601f631b119b61ff7 847B / 847B
-> => sha256:bc38caa0f5b94141276220daaf428892096e4afd24b05668cd188311e00a635f 35.37kB / 35.37kB
-> => sha256:36505266dcc64eeb1010bd2112e6f73981e1a8246e4f6d4e287763b57f101b0b 161B / 161B
-> => sha256:23884877105a7f84a910895cd044061a561385ff6c3c480ee080b7bec0e771 26.69MB / 26.69MB
-> => extracting sha256:23884877105a7f84a910895cd044061a561385ff6c3c480ee080b7bec0e771
-> => extracting sha256:bc38caa0f5b94141276220daaf428892096e4afd24b05668cd188311e00a635f
-> => extracting sha256:2910811b6c4227c2f42aaea9a3dd5f53b1d469f67e2cf7e601f631b119b61ff7
-> => extracting sha256:36505266dcc64eeb1010bd2112e6f73981e1a8246e4f6d4e287763b57f101b0b
-> [2/2] RUN default_user=$(getent passwd 1000 | awk -F ':' '{print $1}') || echo "uid: 1000 does not exist" && default_group=$(getent group 100 | awk -F ':' '{pr
-> => exporting to image
-> => exporting layers
-> => exporting manifest sha256:b239078457df7c75d57a45989cf8d9d08e6fd9dc882a4ede6d4311bc487d80e9
-> => exporting config sha256:6794fa8ae0cc9464b7f3102345237559fb82a377296309b954841b1346cd51db
-> => pushing layers
-> => pushing manifest for swr.cn-north-7.myhuaweicloud.com/notebook_test/my_image:0.0.1@sha256:b239078457df7c75d57a45989cf8d9d08e6fd9dc882a4ede6d4311bc487d80e9
-> [auth] notebook_test/my_image:pull,push token for swr. ....com
*****
*                               Summary Board                               *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* Image Build Time: 4.3s                                                    *
* Repository: swr. ....com/notebook_test/my_image                        *
* Tag: 0.0.1                                                                *
* Compressed Image Size: 25MB                                              *
* SWR Download Command: docker pull swr. ....l.com/notebook_test/my_image:0.0.1 *
*****
(PyTorch-1.8) [ma-user work]$
```

Using ma-cli image df to Obtain Image Building Caches in ModelArts Notebook

Run the **ma-cli image df** command to obtain image creation caches. This command is available only on ModelArts notebook instances.

```
$ ma-cli image df -h
```

Usage: ma-cli image df [OPTIONS]

Query disk usage used by image-building in Notebook.

Example:

```
# Query image disk usage
ma-cli image df
```

Options:

```
-v, --verbose      Show detailed information on disk usage.
-D, --debug        Debug Mode. Shows full stack trace when error occurs.

-h, -H, --help     Show this message and exit.
```

Table 5-31 Parameters

Parameter	Data Type	Mandator y	Description
-v / --verbose	Bool	No	Whether to display detailed information. It is disabled by default.

Example: View all image caches in ModelArts notebook.

```
ma-cli image df
```

```
(PyTorch-1.8) [ma-user work]$ma-cli image df
ID                                RECLAIMABLE  SIZE      LAST ACCESSED
iwrwwsi9pdcjafel1j6d0r918      true         98.50MB
cp52c4q81ud2abu2vp7s35vyt      true         1.04MB
4jbo6v06r2w1575ddq3w8g12e      true         139.68kB
ojd1w5mok71s1nh2cauant051      true         86.86kB
k2jm6g061n5twmz7gmonmqjsh      true         16.55kB
efu5kwgig1ve44fe7smbnrcnh*     true         8.19kB
uzikwqk5taxnslvajm14jrbje*     true         4.10kB
2g8p0qcb014g3qva7ucawkv87*     true         4.10kB
Reclaimable: 99.80MB
Total: 99.80MB
```

Example: Obtain details about the disk space occupied by image caches.

`ma-cli image df --verbose`

```
(Pytorch-1.8) [ma-user work]$ma-cli image df --verbose
ID:          iwrws19pdcjafelij6d0r918
Created at:   2023-03-28 12:23:28.353759532 +0000 UTC
Mutable:     false
Reclaimable: true
Shared:      false
Size:        98.50MB
Description:  pulled from swr. ....com/atelier/ubuntu:18.04@sha256:b5874...a65d84f
Usage count: 1
Last used:   2023-03-28 12:23:28.37337776 +0000 UTC
Type:        regular

ID:          cp52c4q8iud2abu2vp7s35vyt
Parents:     iwrws19pdcjafelij6d0r918
Created at:   2023-03-28 12:23:28.366918223 +0000 UTC
Mutable:     false
Reclaimable: true
Shared:      false
Size:        1.04MB
Description:  pulled from swr. ....com/atelier/ubuntu:18.04@sha256:b5874...e235eea65d84f
Usage count: 1
Last used:   2023-03-28 12:23:28.38560437 +0000 UTC
Type:        regular

ID:          4jb06v06r2w1575ddq3w8g12e
Parents:     k2jpe6g861n5twm27gsoemqjsh
Created at:   2023-03-28 12:23:30.681643727 +0000 UTC
Mutable:     false
Reclaimable: true
Shared:      false
Size:        139.68MB
Description:  mount / from exec /bin/sh -c default_user=$(getent passwd 1000 | awk -F ':' '{print $1}') || echo "uid: 1000 does not exist" && default_group=$(getent
up 100 | awk -F ':' '{print $1}') || echo "gid: 100 does not exist" && if [ ! -z ${default_user} ] && [ ${default_user} != "ma-user" ]; then userdel -r ${defau
user}; fi && if [ ! -z ${default_group} ] && [ ${default_group} != "ma-group" ]; then groupdel -f ${default_group}; fi && groupadd -g 100 ma-group
useradd -d /home/ma-user -m -u 1000 -g 100 -s /bin/bash ma-user && chmod -R 750 /home/ma-user
Usage count: 2
Last used:   2023-03-28 12:25:39.149080471 +0000 UTC
Type:        regular
```

Using ma-cli image prune to Clear Image Building Caches in ModelArts Notebook

Run the **ma-cli image prune** command to clear image creation caches. This command is available only on ModelArts notebook instances.

```
$ ma-cli image prune -h
Usage: ma-cli image prune [OPTIONS]
```

Prune image build cache by image-building in Notebook.

Example:

```
# Prune image build cache
ma-cli image prune
```

Options:

```
-ks, --keep-storage INTEGER Amount of disk space to keep for cache below this limit (in MB) (default: 0).
-kd, --keep-duration TEXT   Keep cache newer than this limit, support second(s), minute(m) and hour(h)
                             (default: 0s).
-v, --verbose                Show more verbose output.
-D, --debug                  Debug Mode. Shows full stack trace when error occurs.
-h, -H, --help               Show this message and exit.
```

Table 5-32 Parameters

Parameter	Data Type	Mand atory	Description
-ks / --keep-storage	Int	No	Size of the cache to be retained, in MB. The default value is 0 , indicating that all caches will be cleared.
-kd / --keep-duration	String	No	Duration in which the caches are to be retained. The unit can be s (second), m (minute), or h (hour). The default value is 0s , indicating that all caches will be cleared.

Parameter	Data Type	Mandatory	Description
-v / --verbose	Bool	No	Whether to display detailed information. It is disabled by default.

Example: Retain 1 MB of image caches not to be cleared.

```
ma-cli image prune -ks 1
```

```
(PyTorch-1.8) [ma-user work]$ma-cli image prune -ks 1
ID                                     RECLAIMABLE  SIZE  LAST ACCESSED
uzikwqk5taxnslvajm14jrbje*          true         4.10kB
4jbo6v06r2w1575ddq3w8g12e          true        139.68kB
k2jm6g061n5twmz7gmonmqjsh          true         16.55kB
ojdjw5mok71s1nh2cauant051          true         86.86kB
cp52c4q81ud2abu2vp7sj5vyt          true         1.04MB
iwrwrsi9pdcjafe1ij6d0r918          true         98.50MB
Total: 99.79MB
```

Using ma-cli image register to Register an SWR Image with ModelArts Image Management

After an image is debugged, run the **ma-cli image register** command to register it with ModelArts image management so that it can be used in ModelArts.

```
$ma-cli image register -h
Usage: ma-cli image register [OPTIONS]
```

Register image to ModelArts.

Example:

```
# Register image into ModelArts service
ma-cli image register --swr-path=xx
```

```
# Share SWR image to DLI service
ma-cli image register -swr xx -td
```

```
# Register image into ModelArts service and specify architecture to be 'AARCH64'
ma-cli image register --swr-path=xx --arch AARCH64
```

Options:

```
-swr, --swr-path TEXT          SWR path without swr endpoint, eg:organization/image:tag. [required]
-a, --arch [X86_64|AARCH64]   Image architecture (default: X86_64).
-s, --service [NOTEBOOK|MODELBOX]
```

Services supported by this image(default NOTEBOOK).

```
-rs, --resource-category [CPU|GPU|ASCEND]
```

The resource category supported by this image (default: CPU and GPU).

```
-wi, --workspace-id TEXT       The workspace to register this image (default: "0").
```

```
-v, --visibility [PUBLIC|PRIVATE]
```

PUBLIC: every user can use this image. PRIVATE: only image owner can use this

image (Default: PRIVATE).

```
-td, --to-dli                  Register swr image to DLI, which will share SWR image to DLI service.
```

```
-d, --description TEXT         Image description (default: "").
```

```
-C, --config-file PATH         Configure file path for authorization.
```

```
-D, --debug                    Debug Mode. Shows full stack trace when error occurs.
```

```
-P, --profile TEXT             CLI connection profile to use. The default profile is "DEFAULT".
```

```
-h, -H, --help                Show this message and exit.
```

Table 5-33 Parameters

Parameter	Data Type	Mandatory	Description
-swr / --swr-path	String	Yes	SWR path to the image to be registered.
-a / --arch	String	No	Architecture of the image to be registered. The value can be X86_64 or AARCH64 . The default value is X86_64 .
-s / --service	String	No	Service type of the image to be registered. The value can be NOTEBOOK or MODELBOX . The default value is NOTEBOOK . You can also specify multiple values, for example, -s NOTEBOOK -s MODELBOX .
-rs / --resource-category	String	No	Resource type that can be used by the image to be registered. The default values are CPU and GPU .
-wi / --workspace-id	String	No	Workspace to which the image is to be registered. The default workspace ID is 0 .
-v / --visibility	Bool	No	Visibility of the image to be registered. The value can be PRIVATE (visible only to the image owner) or PUBLIC (visible to all users). The default value is PRIVATE .
-td / --to-dli	Bool	No	Whether to register the image with DLI.
-d / --description	String	No	Description of the image. By default, this parameter is left blank.

Example: Register an SWR image with ModelArts.

```
ma-cli image register --swr-path=xx
```

```
(PyTorch-1.8) [ma-user work]$ma-cli image register --swr-path=swr.cn-north-4.maas.huaweicloud.com/notebook-test/my_image:0.0.1
You are now in a notebook or devcontainer and cannot use 'ImageManagement.debug' to check your image. If you need to debug it, please use a workstation
[ OK ] Successfully registered this image and image information is
{
  "arch": "x86_64",
  "create_at": 1680006812157,
  "dev_services": [
    "NOTEBOOK",
    "SSH"
  ],
  "id": "850a66748",
  "name": "my_image",
  "namespace": "notebook_test",
  "origin": "CUSTOMIZE",
  "resource_categories": [
    "GPU",
    "CPU"
  ],
  "service_type": "UNKNOWN",
  "size": 26735097,
  "status": "ACTIVE",
  "swr_path": "swr.cn-north-4.maas.huaweicloud.com/notebook-test/my_image:0.0.1",
  "tag": "0.0.1",
  "tags": [],
  "type": "DEDICATED",
  "update_at": 1680006812157,
  "visibility": "PRIVATE",
  "workspace_id": "0"
```

Using ma-cli image unregister to Deregister an Image

Run the **ma-cli image unregister** command to deregister an image from ModelArts.

```
$ ma-cli image unregister -h
Usage: ma-cli image unregister [OPTIONS]

Unregister image from ModelArts.

Example:

# Unregister image
ma-cli image unregister --image-id=xx

# Unregister image and delete it from swr
ma-cli image unregister --image-id=xx -d

Options:
  -i, --image-id TEXT    Unregister image details by image id. [required]
  -d, --delete-swr-image Delete the image from swr.
  -C, --config-file PATH Configure file path for authorization.
  -D, --debug            Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT     CLI connection profile to use. The default profile is "DEFAULT".
  -h, -H, --help        Show this message and exit.
```

Table 5-34 Parameters

Parameter	Data Type	Mandatory	Description
-i / -image-id	String	Yes	ID of the image to be deregistered.
-d / --delete-swr-image	Bool	No	Whether to delete a deregistered SWR image. It is disabled by default.

Debugging an SWR Image on an ECS

ma-cli allows you to debug an SWR image on an ECS to determine whether the image can be used in a ModelArts development workspace.

```
ma-cli image debug -h
Usage: ma-cli image debug [OPTIONS]

Debug SWR image as a Notebook in ECS.

Example:

# Debug cpu notebook image
ma-cli image debug --swr-path=xx --service=NOTEBOOK --region=

# Debug gpu notebook image
ma-cli image debug --swr-path=xx --service=NOTEBOOK --region= --gpu

Options:
  -swr, --swr-path TEXT          SWR path without SWR endpoint, eg:organization/image:tag. [required]
  -r, --region TEXT              Region name. [required]
  -s, --service [NOTEBOOK|MODELBOX]
                                   Services supported by this image(default NOTEBOOK).
  -a, --arch [X86_64|AARCH64]    Image architecture(default X86_64).
  -g, --gpu                      Use all gpus to debug.
  -D, --debug                    Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT             CLI connection profile to use. The default profile is "DEFAULT".
  -h, -H, --help                Show this message and exit.
```

Table 5-35 Parameters

Parameter	Data Type	Mandatory	Description
-swr / --swr-path	String	Yes	SWR path to the image to be debugged.
-r / --region	String	Yes	Region where the image to be debugged is located.
-s / --service	String	No	Service type of the image to be debugged. The value can be NOTEBOOK or MODELBOX . The default value is NOTEBOOK .
-a / --arch	String	No	Architecture of the image to be debugged. The value can be X86_64 or AARCH64 . The default value is X86_64 .
-g / --gpu	Bool	No	Whether to use GPUs for debugging. It is disabled by default.

5.9.6 ma-cli ma-job Commands for Training Jobs

Run the **ma-cli ma-job** command to submit training jobs, obtain training job logs, events, used AI engines, and resource specifications, and stop training jobs.

```
$ ma-cli ma-job -h
Usage: ma-cli ma-job [OPTIONS] COMMAND [ARGS]...

ModelArts job submission and query job details.
```

Options:
-h, -H, --help Show this message and exit.

Commands:
delete Delete training job by job id.
get-engine Get job engines.
get-event Get job running event.
get-flavor Get job flavors.
get-job Get job details.
get-log Get job log details.
get-pool Get job engines.
stop Stop training job by job id.
submit Submit training job.

Table 5-36 Commands supported by training jobs

Command	Description
get-job	Obtain ModelArts training jobs and their details.
get-log	Obtain runtime logs of a ModelArts training job.
get-engine	Obtain ModelArts AI engines for training.
get-event	Obtain ModelArts training job events.
get-flavor	Obtain ModelArts resource specifications for training.
get-pool	Obtain ModelArts resource pools dedicated for training.
stop	Stop a ModelArts training job.
submit	Submit a ModelArts training job.
delete	Delete a training job with a specified job ID.

Using ma-cli ma-job get-job to Obtain a ModelArts Training Job

Run the **ma-cli ma-job get-job** command to obtain training jobs or details about a specific job.

```
$ ma-cli ma-job get-job -h
Usage: ma-cli ma-job get-job [OPTIONS]

Get job details.

Example:

# Get train job details by job name
ma-cli ma-job get-job -n ${job_name}

# Get train job details by job id
ma-cli ma-job get-job -i ${job_id}

# Get train job list
ma-cli ma-job get-job --page-size 5 --page-num 1

Options:

-i, --job-id TEXT          Get training job details by job id.
-n, --job-name TEXT       Get training job details by job name.
-pn, --page-num INTEGER   Specify which page to query. [x>=1]
-ps, --page-size INTEGER RANGE The maximum number of results for this query. [1<=x<=50]
```

-v, --verbose	Show detailed information about training job details.
-C, --config-file TEXT	Configure file path for authorization.
-D, --debug	Debug Mode. Shows full stack trace when error occurs.
-P, --profile TEXT	CLI connection profile to use. The default profile is "DEFAULT".
-h, -H, --help	Show this message and exit.

Table 5-37 Parameters

Parameter	Data Type	Mandato ry	Description
-i / --job-id	String	No	ID of the job whose details are to be obtained.
-n / --job-name	String	No	Name of the job to be queried or name keyword used to filter training jobs.
-pn / --page-num	Int	No	Page number. The default value is 1 .
-ps / --page-size	Int	No	Number of training jobs displayed on each page. The default value is 10 .
-v / --verbose	Bool	No	Whether to display detailed information. It is disabled by default.

- Example: Obtain a training job with a specified job ID.

```
ma-cli ma-job get-job -i b63e90xxx
```

```
(PyTorch-1.4) [ma-user work]$ma-cli ma-job get-job -i b63e90ba-91
```

id	name	status	user_name	duration	create_time	start_time	descripti
b63e90ba-91	workFlow_created_job_ed3a962f-5438-4a99-9a19-c97ce88c48b8	Completed	ei_modela	00h:01m:16s	2023-03-29 03:41:21	2023-03-29 03:41:30	

- Example: Filter training jobs by job name keyword **auto**.

```
ma-cli ma-job get-job -n auto
```

```
(PyTorch-1.4) [ma-user work]$ma-cli ma-job get-job -n auto
```

inde	id	name	status	user_name	duration	create_time	start_time
1	9b495c		Completed		00h:01m:31s	2023-03-29 07:03:08	2023-03-29 07:05:20
2	af2147f5-		Terminated		00h:10m:49s	2023-03-29 06:52:16	2023-03-29 06:52:32
3	2c1855b1-		Failed		00h:37m:29s	2023-03-29 03:22:31	2023-03-29 03:22:58
4	4525b3c9-		Failed		00h:00m:01s	2023-03-29 03:19:41	2023-03-29 03:19:49
5	4234455d-		Terminated		00h:00m:00s	2023-03-29 02:25:18	N/A
6	9810ae49-		Terminated		00h:09m:06s	2023-03-29 02:19:49	2023-03-29 02:20:13
7	90c7de89-		Abnormal		00h:00m:00s	2023-03-29 01:43:18	N/A
8	fc740dc5-		Terminated		00h:00m:00s	2023-03-29 01:22:19	N/A
9	5d16fdfe-		Terminated		00h:00m:00s	2023-03-29 01:11:26	N/A
10	3737e56d-		Completed		00h:05m:59s	2023-03-29 00:59:28	2023-03-29 01:04:20

Using ma-cli ma-job submit to Submit a ModelArts Training Job

Run the **ma-cli ma-job submit** command to submit a ModelArts training job.

When running this command, use the **YAML_FILE** parameter to specify the path to the configuration file of the target job. If this parameter is not specified, the

configuration file is empty. The configuration file is in YAML format, and its parameters are values of **OPTIONS** in the command. If you specify both the **YAML_FILE** and the **OPTIONS** parameters, the **OPTIONS** value will overwrite the same items in the configuration file.

```
$ma-cli ma-job submit -h
Usage: ma-cli ma-job submit [OPTIONS] [YAML_FILE]...

Submit training job.

Example:

ma-cli ma-job submit --code-dir obs://your_bucket/code/
                    --boot-file main.py
                    --framework-type PyTorch
                    --working-dir /home/ma-user/modelarts/user-job-dir/code
                    --framework-version pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64
                    --data-url obs://your_bucket/dataset/
                    --log-url obs://your_bucket/logs/
                    --train-instance-type modelarts.vm.cpu.8u
                    --train-instance-count 1

Options:
  --name TEXT                Job name.
  --description TEXT          Job description.
  --image-url TEXT            Full swr custom image path.
  --uid TEXT                  Uid for custom image (default: 1000).
  --working-dir TEXT          ModelArts training job working directory.
  --local-code-dir TEXT       ModelArts training job local code directory.
  --user-command TEXT         Execution command for custom image.
  --pool-id TEXT              Dedicated pool id.
  --train-instance-type TEXT   Train worker specification.
  --train-instance-count INTEGER Number of workers.
  --data-url TEXT              OBS path for training data.
  --log-url TEXT              OBS path for training log.
  --code-dir TEXT             OBS path for source code.
  --output TEXT               Training output parameter with OBS path.
  --input TEXT                Training input parameter with OBS path.
  --env-variables TEXT         Env variables for training job.
  --parameters TEXT           Training job parameters (only keyword parameters are supported).
  --boot-file TEXT            Training job boot file path behinds `code_dir`.
  --framework-type TEXT       Training job framework type.
  --framework-version TEXT     Training job framework version.
  --workspace-id TEXT         The workspace where you submit training job(default "0")
  --policy [regular|economic|turbo|auto]
                               Training job policy, default is regular.
  --volumes TEXT              Information about the volumes attached to the training job.
  -q, --quiet                 Exit without waiting after submit successfully.
  -C, --config-file PATH      Configure file path for authorization.
  -D, --debug                 Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT           CLI connection profile to use. The default profile is "DEFAULT".
  -H, -h, --help              Show this message and exit.
```

Table 5-38 Parameters

Parameter	Data Type	Mandatory	Description
YAML_FILE	String	No	Configuration file of a training job. If this parameter is not specified, the configuration file is empty.

Parameter	Data Type	Mandatory	Description
--code-dir	String	Yes	OBS path to the training source code.
--data-url	String	Yes	OBS path to the training data.
--log-url	String	Yes	OBS path to training logs.
--train-instance-count	String	Yes	Number of compute nodes in a training job. The default value is 1 , indicating a standalone node.
--boot-file	String	No	Boot file specified when you use a preset command to submit a training job. This parameter can be omitted when you use a custom image or a custom command to submit a training job.
--name	String	No	Name of a training job.
--description	String	No	Description of a training job.
--image-url	String	No	SWR URL of a custom image, which is in the format of "organization/image_name:tag".
--uid	String	No	UID of the custom image. The default value is 1000 .
--working-dir	String	No	Work directory where an algorithm is executed.
--local-code-dir	String	No	Local directory of the training container to which the algorithm code directory is downloaded.
--user-command	String	No	Command for executing a custom image. The directory must be under /home . When code-dir is prefixed with file:// , this parameter does not take effect.
--pool-id	String	No	Resource pool ID selected for a training job. You can log in to the ModelArts console, choose Dedicated Resource Pools in the navigation pane on the left, and view the resource pool ID in the dedicated resource pool list.
--train-instance-type	String	No	Resource flavor selected for a training job.

Parameter	Data Type	Mandatory	Description
--output	String	No	Training output. After this parameter is specified, the training job will upload the output directory of the training container corresponding to the specified output parameter in the training script to a specified OBS path. To specify multiple parameters, use --output output1=obs://bucket/output1 --output output2=obs://bucket/output2 .
--input	String	No	Training input. After this parameter is specified, the training job will download the data from OBS to the training container and transfer the data storage path to the training script through the specified parameter. To specify multiple parameters, use --input data_path1=obs://bucket/data1 --input data_path2=obs://bucket/data2 .
--env-variables	String	No	Environment variables input during training. To specify multiple parameters, use --env-variables ENV1=env1 --env-variables ENV2=env2 .
--parameters	String	No	Training input parameters. To specify multiple parameters, use --parameters "--epoch 0 --pretrained" .
--framework-type	String	No	Framework type selected for a training job.
--framework-version	String	No	Framework version selected for a training job.
-q / --quiet	Bool	No	Whether to exit directly without printing the job status synchronously after a training job is submitted.
--workspace-id	String	No	Workspace where a training job is deployed. The default value is 0 .
--policy	String	No	Training resource flavor mode. The options are regular , economic , turbo , and auto .

Parameter	Data Type	Mandatory	Description
--volumes	String	No	EFS disks to be mounted. To specify multiple parameters, use --volumes . "local_path=/xx/yy/zz;read_only=false;nfs_server_path=xxx.xxx.xxx.xxx:/ " -volumes "local_path=/xxx/yyy/zzz;read_only=false;nfs_server_path=xxx.xxx.xxx.xxx:/"

Example: Submitting a Training Job Based on a Preset ModelArts Image

Submit a training job by specifying the **OPTIONS** parameter.

```
ma-cli ma-job submit --code-dir obs://your-bucket/mnist/code/ \  
  --boot-file main.py \  
  --framework-type PyTorch \  
  --working-dir /home/ma-user/modelarts/user-job-dir/code \  
  --framework-version pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64 \  
  --data-url obs://your-bucket/mnist/dataset/MNIST/ \  
  --log-url obs://your-bucket/mnist/logs/ \  
  --train-instance-type modelarts.vm.cpu.8u \  
  --train-instance-count 1 \  
  -q
```

Example of **train.yaml** using a preset image:

```
# Example .ma/train.yaml (preset image)  
# pool_id: pool_xxxx  
train-instance-type: modelarts.vm.cpu.8u  
train-instance-count: 1  
data-url: obs://your-bucket/mnist/dataset/MNIST/  
code-dir: obs://your-bucket/mnist/code/  
working-dir: /home/ma-user/modelarts/user-job-dir/code  
framework-type: PyTorch  
framework-version: pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64  
boot-file: main.py  
log-url: obs://your-bucket/mnist/logs/  
  
##[Optional] Uncomment to set uid when use custom image mode  
uid: 1000  
  
##[Optional] Uncomment to upload output file/dir to OBS from training platform  
output:  
  - name: output_dir  
    obs_path: obs://your-bucket/mnist/output1/  
  
##[Optional] Uncomment to download input file/dir from OBS to training platform  
input:  
  - name: data_url  
    obs_path: obs://your-bucket/mnist/dataset/MNIST/  
  
##[Optional] Uncomment pass hyperparameters  
parameters:  
  - epoch: 10  
  - learning_rate: 0.01  
  - pretrained:
```

```
##[Optional] Uncomment to use dedicated pool
pool_id: pool_xxxx

##[Optional] Uncomment to use volumes attached to the training job
volumes:
- efs:
    local_path: /xx/yy/zz
    read_only: false
    nfs_server_path: xxx.xxx.xxx.xxx:/
```

Example: Using a Custom Image to Create a Training Job

Submit a training job by specifying the **OPTIONS** parameter.

```
ma-cli ma-job submit --image-url atelier/pytorch_1_8:pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-
x86_64-20220926104358-041ba2e \
    --code-dir obs://your-bucket/mnist/code/ \
    --user-command "export LD_LIBRARY_PATH=/usr/local/cuda/compat:$LD_LIBRARY_PATH && cd /home/ma-user/modelarts/user-job-dir/code && /home/ma-user/anaconda3/envs/PyTorch-1.8/bin/
python main.py" \
    --data-url obs://your-bucket/mnist/dataset/MNIST/ \
    --log-url obs://your-bucket/mnist/logs/ \
    --train-instance-type modelarts.vm.cpu.8u \
    --train-instance-count 1 \
    -q
```

Example of **train.yaml** using a custom image:

```
# Example .ma/train.yaml (custom image)
image-url: atelier/pytorch_1_8:pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-
x86_64-20220926104358-041ba2e
user-command: export LD_LIBRARY_PATH=/usr/local/cuda/compat:$LD_LIBRARY_PATH && cd /home/ma-
user/modelarts/user-job-dir/code && /home/ma-user/anaconda3/envs/PyTorch-1.8/bin/python main.py
train-instance-type: modelarts.vm.cpu.8u
train-instance-count: 1
data-url: obs://your-bucket/mnist/dataset/MNIST/
code-dir: obs://your-bucket/mnist/code/
log-url: obs://your-bucket/mnist/logs/

##[Optional] Uncomment to set uid when use custom image mode
uid: 1000

##[Optional] Uncomment to upload output file/dir to OBS from training platform
output:
- name: output_dir
  obs_path: obs://your-bucket/mnist/output1/

##[Optional] Uncomment to download input file/dir from OBS to training platform
input:
- name: data_url
  obs_path: obs://your-bucket/mnist/dataset/MNIST/

##[Optional] Uncomment pass hyperparameters
parameters:
- epoch: 10
- learning_rate: 0.01
- pretrained:

##[Optional] Uncomment to use dedicated pool
pool_id: pool_xxxx

##[Optional] Uncomment to use volumes attached to the training job
volumes:
- efs:
    local_path: /xx/yy/zz
    read_only: false
    nfs_server_path: xxx.xxx.xxx.xxx:/
```

Using ma-cli ma-job get-log to Obtain ModelArts Training Job Logs

Run the **ma-cli ma-job get-log** command to obtain ModelArts training job logs.

```
$ ma-cli ma-job get-log -h
Usage: ma-cli ma-job get-log [OPTIONS]
```

Get job log details.

Example:

```
# Get job log by job id
ma-cli ma-job get-log --job-id ${job_id}
```

Options:

```
-i, --job-id TEXT      Get training job details by job id. [required]
-t, --task-id TEXT     Get training job details by task id (default "worker-0").
-C, --config-file TEXT Configure file path for authorization.
-D, --debug            Debug Mode. Shows full stack trace when error occurs.
-P, --profile TEXT     CLI connection profile to use. The default profile is "DEFAULT".
-h, -H, --help        Show this message and exit.
```

Parameter	Data Type	Mandatory	Description
-i / --job-id	String	Yes	ID of the job whose logs are to be obtained.
-t / --task-id	String	No	ID of the task whose logs are to be obtained. The default value is work-0 .

Example: Obtain logs of a specified training job.

```
ma-cli ma-job get-log --job-id b63e90baxxx
```

```
(PyTorch 1.4) [ma-user work]$ma-cli ma-job get-log --job-id b63e90ba
time="2023-03-29T11:41:26+08:00" level=info msg="init logger successful" file="init.go:55" Command=bootstrap/init Component=ma-training-toolkit Platform=ModelArts-Service
time="2023-03-29T11:41:26+08:00" level=info msg="current user 1000:1000" file="init.go:57" Command=bootstrap/init Component=ma-training-toolkit Platform=ModelArts-Service
time="2023-03-29T11:41:27+08:00" level=info msg="report even" file="init.go:61" Command=bootstrap/init Component=ma-training-toolkit Platform=ModelArts-Service
time="2023-03-29T11:41:27+08:00" level=info msg="init command" file="init.go:81" Command=bootstrap/init Component=ma-training-toolkit Platform=ModelArts-Service
time="2023-03-29T11:41:27+08:00" level=info msg="acc is alive" file="init.go:83" Command=bootstrap/init Component=ma-training-toolkit Platform=ModelArts-Service
time="2023-03-29T11:41:27+08:00" level=info msg="[init] tool" file="init.go:85" Command=bootstrap/init Component=ma-training-toolkit Platform=ModelArts-Service
time="2023-03-29T11:41:27+08:00" level=info msg="[init] runn" file="init.go:87" Command=bootstrap/init Component=ma-training-toolkit Platform=ModelArts-Service
time="2023-03-29T11:41:27+08:00" level=info msg="[init] ip o" file="init.go:89" Command=bootstrap/init Component=ma-training-toolkit Platform=ModelArts-Service
time="2023-03-29T11:41:27+08:00" level=info msg="local dir = " file="init.go:91" Command=bootstrap/init Component=ma-training-toolkit Platform=ModelArts-Service
time="2023-03-29T11:41:27+08:00" level=info msg="obs dir = s" file="init.go:93" Command=bootstrap/init Component=ma-training-toolkit Platform=ModelArts-Service
time="2023-03-29T11:41:27+08:00" level=info msg="num of workers = 8" file="upload.go:214" Command=obs/upload Component=ma-training-toolkit Platform=ModelArts-Service Task=
time="2023-03-29T11:41:27+08:00" level=info msg="start the periodic upload task, upload Period = 5 seconds " file="upload.go:220" Command=obs/upload Component=ma-training-toolkit Platform=ModelArts-Service Task=
time="2023-03-29T11:41:27+08:00" level=info msg="report event DetectStart success" file="event.go:63" Command=report Component=ma-training-toolkit Platform=ModelArts-Service
```

Using ma-cli ma-job get-event to Obtain ModelArts Training Job Events

Run the **ma-cli ma-job get-event** command to obtain ModelArts training job events.

```
$ ma-cli ma-job get-event -h
Usage: ma-cli ma-job get-event [OPTIONS]
```

Get job running event.

Example:

```
# Get training job running event
ma-cli ma-job get-event --job-id ${job_id}
```

Options:

```
-i, --job-id TEXT      Get training job event by job id. [required]
-C, --config-file TEXT Configure file path for authorization.
```

-D, --debug Debug Mode. Shows full stack trace when error occurs.
-P, --profile TEXT CLI connection profile to use. The default profile is "DEFAULT".
-H, -h, --help Show this message and exit.

Parameter	Data Type	Mandatory	Description
-i / --job-id	String	Yes	ID of the job whose events are to be obtained.

Example: Obtain events of a specified training job.

```
ma-cli ma-job get-event --job-id b63e90baxxx
```

STAT	INFO	TIME
[]	Training job completed.	2023-03-29T11:4 2:47+08:00
[OK]		
[0m]	[worker-0][time used: 0.136s] Upload training output(parameter name: output_url) finished.	2023-03-29T11:4 2:42+08:00
[]		
[OK]		
[0m]	[worker-0] Training output(parameter name: output_url) Uploading.	2023-03-29T11:4 2:42+08:00
[]		
[OK]		
[0m]	[Job: modelarts-job-b63e90ba-...] ExecuteAction: Start to execute action CompleteJob	2023-03-29T11:4 2:42+08:00
[]		
[OK]		
[0m]	[worker-0] Training finished. Exit code 0.	2023-03-29T11:4 2:40+08:00
[]		
[OK]		
[0m]		
[]	[worker-0] training started.	2023-03-29T11:4 1:38+08:00
[]		

Using ma-cli ma-job get-engine to Obtain the AI Engines Used by ModelArts Training Jobs

Run the **ma-cli ma-job get-engine** command to obtain the AI engines used by ModelArts training jobs.

```
$ ma-cli ma-job get-engine -h
Usage: ma-cli ma-job get-engine [OPTIONS]
```

Get job engine info.

Example:

```
# Get training job engines
ma-cli ma-job get-engine
```

Options:

-v, --verbose Show detailed information about training engines.
-C, --config-file TEXT Configure file path for authorization.
-D, --debug Debug Mode. Shows full stack trace when error occurs.
-P, --profile TEXT CLI connection profile to use. The default profile is "DEFAULT".
-H, -h, --help Show this message and exit.

Table 5-39 Parameters

Parameter	Data Type	Mandatory	Description
-v / --verbose	Bool	No	Whether to display detailed information. It is disabled by default.

Example: Obtain the AI engines used by training jobs.

```
ma-cli ma-job get-engine
```

```
(PyTorch-1.4) [ma-user work]$ma-cli ma-job get-engine
```

index	engine id	engine name	run user
1	caffe-1.0.0-python2.7	Caffe	
2	horovod-cp36-tf-1.16.2	Horovod	
3	horovod_0.20.0-pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64	Horovod	1102
4	horovod_0.20.0-tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64	Horovod	1102
5	kungfu-0.2.2-tf-1.13.1-python3.6	KungFu	
6	mindspore_1.3.0-cuda_10.1-py_3.7-ubuntu_1804-x86_64	MPI	1102
7	mindspore_1.7.0-cann_5.1.0-py_3.7-euler_2.8.3-aarch64	Ascend-Powered-Engine	1000
8	mindspore_1.8.0-cann_5.1.2-py_3.7-euler_2.8.3-aarch64	Ascend-Powered-Engine	1000
9	mindspore_1.9.0-cann_6.0.0-py_3.7-euler_2.8.3-aarch64	Ascend-Powered-Engine	1000
10	mxnet-1.2.1-python3.6	MXNet	
11	optverse_0.2.0-pygrassland_1.1.0-py_3.7-ubuntu_18.04-x86_64	OR	1000
12	pytorch-cp36-1.0.0	PyTorch	
13	pytorch-cp36-1.3.0	PyTorch	
14	pytorch-cp36-1.4.0	PyTorch	
15	pytorch_1.8.0-cann_5.1.0-py_3.7-euler_2.8.3-aarch64	Ascend-Powered-Engine	1000
16	pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64	PyTorch	1000
17	pytorch_1.8.1-cann_5.1.2-py_3.7-euler_2.8.3-aarch64	Ascend-Powered-Engine	1000
18	pytorch_1.8.1-cann_6.0.0-py_3.7-euler_2.8.3-aarch64	Ascend-Powered-Engine	1000
19	pytorch_1.8.1-cuda_11.1-py_3.7-ubuntu_18.04-x86_64	PyTorch	1000
20	pytorch_1.8.2-cuda_10.2-py_3.7-ubuntu_18.04-x86_64	PyTorch	1000
21	pytorch_1.9.1-cuda_11.1-py_3.7-ubuntu_18.04-x86_64	PyTorch	1000
22	ray-cp36-0.7.4	Ray	

Using ma-cli ma-job get-flavor to Obtain the Resource Flavors Used by ModelArts Training Jobs

Run the **ma-cli ma-job get-flavor** command to obtain the resource flavors used by ModelArts training jobs.

```
$ ma-cli ma-job get-flavor -h
Usage: ma-cli ma-job get-flavor [OPTIONS]
```

Get job flavor info.

Example:

```
# Get training job flavors
ma-cli ma-job get-flavor
```

Options:

```
-t, --flavor-type [CPU|GPU|Ascend]  Type of training job flavor.
-v, --verbose                        Show detailed information about training flavors.
-C, --config-file TEXT               Configure file path for authorization.
-D, --debug                          Debug Mode. Shows full stack trace when error occurs.
-P, --profile TEXT                   CLI connection profile to use. The default profile is "DEFAULT".
-H, -h, --help                       Show this message and exit.
```

Table 5-40 Parameters

Parameter	Data Type	Mandatory	Description
-t / --flavor-type	String	No	Resource flavor type. If this parameter is not specified, all resource flavors are returned by default.
-v / --verbose	Bool	No	Whether to display detailed information. It is disabled by default.

Example: Obtain the resource flavors and types of training jobs.

```
ma-cli ma-job get-flavor
```

```
(PyTorch-1.4) [ma-user work]$ma-cli ma-job get-flavor
```

index	flavor id	flavor name	flavor type
1	modelarts.kat1.8xlarge	Computing NPU(8*Ascend) instance	Ascend
2	modelarts.kat1.xlarge	Computing NPU(Ascend) instance	Ascend
3	modelarts.vm.cpu.2u	Computing CPU(2U) instance	CPU
4	modelarts.vm.cpu.8u	Computing CPU(8U) instance	CPU
5	modelarts.vm.cpu.8u16g.119	Computing CPU(8U) instance	CPU
6	modelarts.vm.v100.large	Computing GPU(V100) instance	GPU
7	modelarts.vm.v100.large.free	Computing GPU(V100) instance	GPU

Using ma-cli ma-job stop to Stop a ModelArts Training Job

Run the **ma-cli ma-job stop** command to stop a training job with a specified job ID.

```
$ ma-cli ma-job stop -h
Usage: ma-cli ma-job stop [OPTIONS]
```

Stop training job by job id.

Example:

```
Stop training job by job id
ma-cli ma-job stop --job-id ${job_id}
```

Options:
-i, --job-id TEXT Get training job event by job id. [required]
-y, --yes Confirm stop operation.
-C, --config-file TEXT Configure file path for authorization.
-D, --debug Debug Mode. Shows full stack trace when error occurs.
-P, --profile TEXT CLI connection profile to use. The default profile is "DEFAULT".
-H, -h, --help Show this message and exit.

Table 5-41 Parameters

Parameter	Data Type	Mandatory	Description
-i / --job-id	String	Yes	ID of a ModelArts training job
-y / --yes	Bool	No	Whether to forcibly stop a training job

Example: Stop a running training job.

```
ma-cli ma-job stop --job-id efd3e2f8xxx
```

5.9.7 ma-cli dli-job Commands for Submitting DLI Spark Jobs

```
$ma-cli dli-job -h
Usage: ma-cli dli-job [OPTIONS] COMMAND [ARGS]...

DLI spark job submission and query job details.

Options:
  -h, -H, --help  Show this message and exit.

Commands:
  get-job      Get DLI spark job details.
  get-log      Get DLI spark log details.
  get-queue    Get DLI spark queues info.
  get-resource Get DLI resources info.
  stop         Stop DLI spark job by job id.
  submit       Submit dli spark batch job.
  upload       Upload local file or OBS object to DLI resources.
```

Table 5-42 Commands for submitting DLI Spark jobs

Command	Description
get-job	Obtain DLI Spark jobs and their details.
get-log	Obtain runtime logs of a DLI Spark job.
get-queue	Obtain DLI queues.
get-resource	Obtain DLI group resources.
stop	Stop a DLI Spark job.
submit	Submit a DLI Spark job.

Command	Description
upload	Upload local files or OBS files to a DLI group.

Using ma-cli dli-job get-job to Obtain a DLI Spark Job

Run the **ma-cli dli-job get-job** command to obtain the DLI Spark jobs or details about a job.

```
ma-cli dli-job get-job -h
Usage: ma-cli dli-job get-job [OPTIONS]
```

Get DLI Spark details.

Example:

```
# Get DLI Spark job details by job name
ma-cli dli-job get-job -n ${job_name}
```

```
# Get DLI Spark job details by job id
ma-cli dli-job get-job -i ${job_id}
```

```
# Get DLI Spark job list
ma-cli dli-job get-job --page-size 5 --page-num 1
```

Options:

```
-i, --job-id TEXT           Get DLI Spark job details by job id.
-n, --job-name TEXT         Get DLI Spark job details by job name.
-pn, --page-num INTEGER RANGE Specify which page to query. [x>=1]
-ps, --page-size INTEGER RANGE The maximum number of results for this query. [x>=1]
-v, --verbose               Show detailed information about DLI Spark job details.
-C, --config-file PATH      Configure file path for authorization.
-D, --debug                 Debug Mode. Shows full stack trace when error occurs.
-P, --profile TEXT           CLI connection profile to use. The default profile is "DEFAULT".
-H, -h, --help              Show this message and exit.
```

Table 5-43 Parameters

Parameter	Data Type	Mandatory	Description
-i / --job-id	String	No	ID of the DLI Spark job whose details are to be obtained.
-n / --job-name	String	No	Name of the DLI Spark job to be queried or name keyword used to filter DLI Spark jobs.
-pn / --page-num	Int	No	Page number. The default value is 1 .
-ps / --page-size	Int	No	Number of jobs displayed on each page. The default value is 20 .
-v / --verbose	Bool	No	Whether to display detailed information. It is disabled by default.

Example: Obtain all DLI Spark jobs.

```
ma-cli dli-job get-job
```

```
(PyTorch-1.8) [ma-user work]$ma-cli dli-job get-job
```

index	id	name	status	queue	sc_type	image
1	15c87f3a-973e	zh	dead	dli_ma_notebook	CUSTOMIZED	notebook
2	656dd759-b04e	zh	dead	dli_ma_notebook	CUSTOMIZED	notebook
3	1a193b8d-335f	zh	dead	dli_ma_notebook	CUSTOMIZED	notebook
4	12fbcc37-8dff	zh	dead	dli_ma_notebook	CUSTOMIZED	notebook
5	794dfd57-bb2e	zh	dead	dli_ma_notebook	CUSTOMIZED	notebook
6	76a3aa43-43bf	zh	dead	dli_ma_notebook	CUSTOMIZED	notebook
7	82856087-5bd3	zh	dead	dli_ma_notebook	CUSTOMIZED	notebook
8	095c0c3f-b0c5	zh	dead	dli_ma_notebook	CUSTOMIZED	notebook
9	a2324e0f-81e1	zh	dead	dli_ma_notebook	CUSTOMIZED	notebook
10	d70717e2-1a3e	zh	dead	dli_ma_notebook	CUSTOMIZED	notebook
11	85358931-99af	zh	dead	dli_ma_notebook	CUSTOMIZED	notebook
12	d5546f21-4305	zh	dead	dli_ma_notebook	CUSTOMIZED	notebook
13	7b3b9fac-0141	zh	dead	dli_ma_notebook	CUSTOMIZED	notebook
14	2495b20b-4c2c	zh	dead	dli_ma_notebook	CUSTOMIZED	notebook
15	59924d24-ef02	zh	dead	dli_ma_notebook	CUSTOMIZED	notebook
16	dab5d88f-cdb5	zh	dead	dli_ma_notebook	CUSTOMIZED	notebook
17	eff42ca1-074e	zh	dead	dli_ma_notebook	CUSTOMIZED	notebook
18	9357a261-72de	zh	dead	dli_ma_notebook	CUSTOMIZED	notebook
19	e5157750-59cc	zh	dead	dli_ma_notebook	CUSTOMIZED	notebook
20	7b273ef2-8e52	zh	dead	dli_ma_notebook	CUSTOMIZED	notebook

Using ma-cli dli-job submit to Submit a DLI Spark Job

Run the **ma-cli dli-job submit** command to submit a DLI Spark job.

When running this command, use the **YAML_FILE** parameter to specify the path to the configuration file of the target job. If this parameter is not specified, the configuration file is empty. The configuration file is in YAML format, and its parameters are values of **OPTIONS** in the command. If you specify both the **YAML_FILE** and the **OPTIONS** parameters, the **OPTIONS** value will overwrite the same items in the configuration file.

Command parameters preview

```
ma-cli dli-job submit -h
Usage: ma-cli dli-job submit [OPTIONS] [YAML_FILE]...
```

Submit DLI Spark job.

Example:

```
ma-cli dli-job submit --name test-spark-from-sdk
                    --file test/sub_dli_task.py
                    --obs-bucket dli-bucket
                    --queue dli_test
                    --spark-version 2.4.5
                    --driver-cores 1
                    --driver-memory 1G
                    --executor-cores 1
                    --executor-memory 1G
                    --num-executors 1
```

Options:

```
--file TEXT          Python file or app jar.
-cn, --class-name TEXT Your application's main class (for Java / Scala apps).
```

```
--name TEXT           Job name.
--image TEXT          Full swr custom image path.
--queue TEXT          Execute queue name.
-obs, --obs-bucket TEXT  DLI obs bucket to save logs.
-sv, --spark-version TEXT  Spark version.
-st, --sc-type [A|B|C]  Compute resource type.
--feature [basic|custom|ai]  Type of the Spark image used by a job (default: basic).
-ec, --executor-cores INTEGER  Executor cores.
-em, --executor-memory TEXT  Executor memory (eg. 2G/2048MB).
-ne, --num-executors INTEGER  Executor number.
-dc, --driver-cores INTEGER  Driver cores.
-dm, --driver-memory TEXT  Driver memory (eg. 2G/2048MB).
--conf TEXT           Arbitrary Spark configuration property (eg. <PROP=VALUE>).
--resources TEXT      Resources package path.
--files TEXT          Files to be placed in the working directory of each executor.
--jars TEXT           Jars to include on the driver and executor class paths.
-pf, --py-files TEXT  Python files to place on the PYTHONPATH for Python apps.
--groups TEXT         User group resources.
--args TEXT           Spark batch job parameter args.
-q, --quiet           Exit without waiting after submit successfully.
-C, --config-file PATH  Configure file path for authorization.
-D, --debug           Debug Mode. Shows full stack trace when error occurs.
-P, --profile TEXT    CLI connection profile to use. The default profile is "DEFAULT".
-H, -h, --help        Show this message and exit.
```

YAML file preview

```
# dli-demo.yaml
name: test-spark-from-sdk
file: test/sub_dli_task.py
obs-bucket: ${your_bucket}
queue: dli_notebook
spark-version: 2.4.5
driver-cores: 1
driver-memory: 1G
executor-cores: 1
executor-memory: 1G
num-executors: 1

## [Optional]
jars:
  - ./test.jar
  - obs://your-bucket/jars/test.jar
  - your_group/test.jar

## [Optional]
files:
  - ./test.csv
  - obs://your-bucket/files/test.csv
  - your_group/test.csv

## [Optional]
python-files:
  - ./test.py
  - obs://your-bucket/files/test.py
  - your_group/test.py

## [Optional]
resources:
  - name: your_group/test.py
    type: pyFile
  - name: your_group/test.csv
    type: file
  - name: your_group/test.jar
    type: jar
  - name: ./test.py
    type: pyFile
  - name: obs://your-bucket/files/test.py
    type: pyFile
```

```
## [Optional]
groups:
- group1
- group2
```

Example of submitting a DLI Spark job by specifying **OPTIONS**:

```
$ ma-cli dli-job submit --name test-spark-from-sdk \
  --file test/sub_dli_task.py \
  --obs-bucket ${your_bucket} \
  --queue dli_test \
  --spark-version 2.4.5 \
  --driver-cores 1 \
  --driver-memory 1G \
  --executor-cores 1 \
  --executor-memory 1G \
  --num-executors 1
```

Table 5-44 Parameters

Parameter	Data Type	Mandator y	Description
YAML_FILE	String	No	Local path to the configuration file of a DLI Spark job. If this parameter is not specified, the configuration file is empty.
--file	String	Yes	Program entry file. It can be a local file path, an OBS path, or the name of a JAR or PyFile package that has been uploaded to the DLI resource management system.
-cn / --class_name	String	Yes	Java/Spark main class of the batch processing job.
--name	String	No	Specified job name. The value consists of a maximum of 128 characters.
--image	String	No	Path to a custom image in the format of "Organization name/Image name:Image version". This parameter is valid only when feature is set to custom . You can use this parameter with the feature parameter to specify a custom Spark image for running a job.
-obs / --obs-bucket	String	No	OBS bucket for storing a Spark job. Configure this parameter when you need to save jobs. It can also be used as a transit station for submitting local files to resources .
-sv/ --spark-version	String	No	Spark version used by a job.

Parameter	Data Type	Mandatory	Description
-st / `--sc-type	String	No	If the current Spark version is 2.3.2, leave this parameter blank. If the current Spark version is 2.3.3, configure this parameter when feature is set to basic or ai . If this parameter is not specified, the default Spark version 2.3.2 will be used.
--feature	String	No	Job feature, indicating the type of the Spark image used by a job. The default value is basic . • basic: A base Spark image provided by DLI is used. • custom: A custom Spark image is used. • ai: An AI image provided by DLI is used.
--queue	String	No	Queue name. Set this parameter to the name of a created DLI queue. The queue must be of the common type. For details about how to obtain a queue name, see Table 5-46 .
-ec / --executor-cores	String	No	Number of CPU cores of each Executor in a Spark application. This configuration will replace the default setting in sc_type .
-em / --executor-memory	String	No	Executor memory of a Spark application, for example, 2 GB or 2048 MB. This configuration will replace the default setting in sc_type . The unit must be provided. Otherwise, the startup fails.
-ne / --num-executors	String	No	Number of Executors in a Spark application. This configuration will replace the default setting in sc_type .
-dc / --driver-cores	String	No	Number of CPU cores of the Spark application driver. This configuration will replace the default setting in sc_type .
-dm / --driver-memory	String	No	Driver memory of a Spark application, for example, 2 GB or 2048 MB. This configuration will replace the default setting in sc_type . The unit must be provided. Otherwise, the startup fails.
--conf	Array of String	No	batch configuration. For details, see Spark Configuration . To specify multiple parameters, use --conf conf1 --conf conf2 .

Parameter	Data Type	Mandatory	Description
--resources	Array of String	No	Name of a resource package, which can be a local file, OBS path, or a file that has been uploaded to the DLI resource management system. To specify multiple parameters, use --resources resource1 --resources resource2 .
--files	Array of String	No	Name of the file package that has been uploaded to the DLI resource management system. You can also specify an OBS path, for example, obs://Bucket name/Package name . Local files are also supported. To specify multiple parameters, use --files file1 --files file2 .
--jars	Array of String	No	Name of the JAR package that has been uploaded to the DLI resource management system. You can also specify an OBS path, for example, obs://Bucket name/Package name . Local files are also supported. To specify multiple parameters, use --jars jar1 --jars jar2 .
-pf /--python-files	Array of String	No	Name of the PyFile package that has been uploaded to the DLI resource management system. You can also specify an OBS path, for example, obs://Bucket name/Package name . Local files are also supported. To specify multiple parameters, use --python-files py1 --python-files py2 .
--groups	Array of String	No	Resource group name. To specify multiple parameters, use --groups group1 --groups group2 .
--args	Array of String	No	Parameters passed to the main class, that is, program parameters. To specify multiple parameters, use --args arg1 --args arg2 .
-q / --quiet	Bool	No	Whether to exit directly without printing the job status synchronously after a DLI Spark job is submitted.

Example

- Submit a DLI Spark job using **YAML_FILE**.
`$ma-cli dli-job submit dli_job.yaml`

```
(PyTorch-1.4) [ma-user work]$ma-cli dli-job submit ./dli-job.yaml
[ OK ] Current DLI job id is: 01b698b8-9fd6-4a8e-bc3c-6821c6405b14
[ OK ] starting
[ OK ] running
[ OK ] success
[ OK ] Successfully submit DLI spark job [ 01b698b8-9fd6-4a8e-bc3c-6821c6405b14 ].
```

- Submit a DLI Spark job by specifying the **OPTIONS** parameter.

```
$ma-cli dli-job submit --name test-spark-from-sdk \
> --file test/jumpstart-trainingjob-gallery-pytorch-sample.ipynb \
> --queue dli_ma_notebook \
> --spark-version 2.4.5 \
> --driver-cores 1 \
> --driver-memory 1G \
> --executor-cores 1 \
> --executor-memory 1G \
> --num-executors 1
```

```
(PyTorch-1.4) [ma-user work]$ma-cli dli-job submit --name test-spark-from-sdk \
> --file test/jumpstart-trainingjob-gallery-pytorch-sample.ipynb \
> --queue dli_ma_notebook \
> --spark-version 2.4.5 \
> --driver-cores 1 \
> --driver-memory 1G \
> --executor-cores 1 \
> --executor-memory 1G \
> --num-executors 1
[ OK ] Current DLI job id is: ae856c20-e9ae-49ca-8409-7a02652297b8
[ OK ] starting
```

Using ma-cli dli-job get-log to Obtain Run Logs of a DLI Spark Job

Run the **ma-cli dli-job get-log** command to obtain the run logs of a DLI Spark job.

```
$ ma-cli dli-job get-log -h
Usage: ma-cli dli-job get-log [OPTIONS]

Get DLI spark job log details.

Example:

# Get job log by job id
ma-cli dli-job get-log --job-id ${job_id}

Options:
  -i, --job-id TEXT      Get DLI spark job details by job id. [required]
  -C, --config-file TEXT Configure file path for authorization.
  -D, --debug            Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT     CLI connection profile to use. The default profile is "DEFAULT".
  -H, -h, --help        Show this message and exit.
```

Table 5-45 Parameters

Parameter	Data Type	Mandator y	Description
-i / --job-id	String	Yes	ID of the DLI Spark job whose logs are to be obtained.

Example: Obtain the run logs of a specified DLI Spark job.

```
ma-cli dli-job get-log --job-id ${your_job_id}
```

```
(PyTorch-1.8) [ma-user work]$ma-cli dli-job get-log --job-id 7b273ef2-8e5
driver:~$ umask 027
++ id -u
+ myuid=2010
++ id -g
+ mygid=2010
+ set +e
++ getent
+ uidentry
+ set -e
+ '[' -z o
+ SPARK_CL
+ grep SPA
+ sort -t
+ sed 's/[
+ env
+ readarra
+ '[' -n '
+ '[' 3 ==
+ '[' 3 ==
++ python3
+ pyv3='Py
+ export P
+ PYTHON_V
+ export P
+ PYSPARK
+ export P
+ PYSPARK
+ '[' -z x
+ SPARK_CL
+ '[' -z '
+ case "$1
+ shift 1
+ CMD=("$S
+ '[' true
+ '[' true
+ '[' -z x
+ '[' -z x
+ exec /us
oy,PythonR
++ tee -a
++ tee -a
++ sed -u -e 's/\[[0-9;]*m//g' -e 's/\x1b//g'
```

Using ma-cli dli-job get-queue to Obtain a DLI Queue

Run the **ma-cli dli-job get-queue** command to obtain a DLI queue.

```
ma-cli dli-job get-queue -h
Usage: ma-cli dli-job get-queue [OPTIONS]

Get DLI queues info.

Example:

# Get DLI queue details by queue name
ma-cli dli-job get-queue --queue-name $queue_name}

Options:
-pn, --page-num INTEGER RANGE  Specify which page to query. [x>=1]
-ps, --page-size INTEGER RANGE  The maximum number of results for this query. [x>=1]
-n, --queue-name TEXT           Get DLI queue details by queue name.
-t, --queue-type [sql|general|all]
                                DLI queue type (default "all").
-tags, --tags TEXT              Get DLI queues by tags.
-C, --config-file PATH         Configure file path for authorization.
-D, --debug                    Debug Mode. Shows full stack trace when error occurs.
-P, --profile TEXT             CLI connection profile to use. The default profile is "DEFAULT".
-H, -h, --help                Show this message and exit.
```

Table 5-46 Parameters

Parameter	Data Type	Mandato ry	Description
-n / --queue-name	String	No	Name of the DLI queue to be obtained.

Parameter	Data Type	Mandatory	Description
-t / --queue-type	String	No	Type of the DLI queue to be obtained. The value can be sql , general , or all . The default value is all .
-tags / --tags	String	No	Tags of the DLI queue to be obtained.
-pn / --page-num	Int	No	DLI queue page number. The default value is 1 .
-ps / --page-size	Int	No	Number of DLI queues displayed on each page. The default value is 20 .

Example: Query information about the queue named **dli_ma_notebook**.

```
ma-cli dli-job get-queue --queue-name dli_ma_notebook
```

```
(PyTorch-1.8) [ma-user work]$ ma-cli dli-job get-queue --queue-name dli_ma_notebook
{'chargingMode': 1,
 'create_time': 1668585417422,
 'cuCount': 16,
 'cu_spec': 16,
 'description': '',
 'enterprise_project_id': '0',
 'is_success': True,
 'message': '',
 'owner': '...',
 'queueName': 'dli_ma_notebook',
 'queueType': 'general',
 'queue_id': 8...,
 'resource_id': '242e7af8-c9d1...',
 'resource_mode': 1,
 'resource_type': 'container',
 'support_spark_versions': ['2.3.2', '2.4.5', '3.1.1']}
```

Using ma-cli dli-job get-resource to Obtain DLI Group Resources

Run the **ma-cli dli-job get-resource** command to obtain details about DLI resources, such as the resource name and resource type.

```
$ ma-cli dli-job get-resource -h
Usage: ma-cli dli-job get-resource [OPTIONS]

Get DLI resource info.

Example:

# Get DLI resource details by resource name
ma-cli dli-job get-resource --resource-name ${resource_name}

Options:
  -n, --resource-name TEXT      Get DLI resource details by resource name.
  -k, --kind [jar|pyFile|file|modelFile]
                                DLI resources type.
  -g, --group TEXT              Get DLI resources by group.
```

-tags, --tags TEXT	Get DLI resources by tags.
-C, --config-file TEXT	Configure file path for authorization.
-D, --debug	Debug Mode. Shows full stack trace when error occurs.
-P, --profile TEXT	CLI connection profile to use. The default profile is "DEFAULT".
-H, -h, --help	Show this message and exit.

Table 5-47 Parameters

Parameter	Data Type	Mandatory	Description
-n / --resource-name	String	No	Resource name of the DLI group resources to be queried.
-k / --kind	String	No	Resource type of the DLI group resources to be queried. The type can be JAR, PyFile, file, or modelFile.
-g / --group	String	No	Group name of the DLI group resources to be queried.
-tags / --tags	String	No	Tags of the DLI group resources to be queried.

Example: Obtain information about all DLI group resources.

```
ma-cli dli-job get-resource
```

```
(PyTorch-1.4) [ma-user work]$ma-cli dli-job get-resource
{'groups': [{'create_time': 1679561988580,
  'details': [{'create_time': 1679561988692,
    'owner': 'ei_...',
    'resource_name': 'Untitled.ipynb',
    'resource_type': 'file',
    'status': 'READY',
    'underlying_name': 'Untitled.ipynb',
    'update_time': 1679561989683}],
  'group_name': 'mrn',
  'is_async': False,
  'owner': 'ei_...',
  'resources': ['Untitled.ipynb'],
  'status': 'READY',
  'update_time': 1679561989683},
{'create_time': 1679561437096,
  'details': [{'create_time': 1679561437233,
    'owner': 'ei_...',
    'resource_name': 'jumpstart-trainingjob-gallery-pytorch-sample.ipynb',
    'resource_type': 'file',
    'status': 'READY',
    'underlying_name': 'jumpstart-trainingjob-gallery-pytorch-sample.ipynb',
    'update_time': 1679561438810},
{'create_time': 1679561929606,
  'owner': 'ei_...',
  'resource_name': 'Untitled.ipynb',
  'resource_type': 'file',
  'status': 'READY',
  'underlying_name': 'Untitled.ipynb',
  'update_time': 1679561930312}],
  'group_name': 'test',
  'is_async': False,
  'owner': 'ei_...',
  'resources': ['jumpstart-trainingjob-gallery-pytorch-sample.ipynb',
    'Untitled.ipynb'],
  'status': 'READY',
  'update_time': 1679561930312}],
'modules': [{'create_time': 1560249470326,
  'description': '',
  'module_name': 'sys.dli.test',
  'module_type': 'jar',
  'resources': [],
  'status': 'READY',
  'update_time': 1560249470339},
{'create_time': 1564118513494,
  'description': '',
  'module_name': 'sys.dli.module',
  'module_type': 'jar',
  'resources': ['spark-examples 2.11-2.1.0.luxor.jar']}]}
```

Using ma-cli dli-job upload to Upload Files to a DLI Group

Run the **ma-cli dli-job upload** command to upload local files or OBS files to a DLI group.

```
$ ma-cli dli-job upload -h
Usage: ma-cli dli-job upload [OPTIONS] PATHS...
```

Upload DLI resource.

Tips: --obs-path is need when upload local file.

Example:

Upload an OBS path to DLI resource

```
ma-cli dli-job upload obs://your-bucket/test.py -g test-group --kind pyFile
```

Upload a local path to DLI resource

```
ma-cli dli-job upload ./test.py -g test-group -obs ${your-bucket} --kind pyFile
```

Upload local path and OBS path to DLI resource

```
ma-cli dli-job upload ./test.py obs://your-bucket/test.py -g test-group -obs ${your-bucket}
```

Options:

-k, --kind [jar|pyFile|file] DLI resources type.
-g, --group TEXT DLI resources group.
-tags, --tags TEXT DLI resources tags, follow --tags `key1`=`value1`.
-obs, --obs-bucket TEXT OBS bucket for upload local file.
-async, --is-async whether to upload resource packages in asynchronous mode. The default value is False.
-C, --config-file TEXT Configure file path for authorization.
-D, --debug Debug Mode. Shows full stack trace when error occurs.
-P, --profile TEXT CLI connection profile to use. The default profile is "DEFAULT".
-H, -h, --help Show this message and exit.

Table 5-48 Parameters

Parameter	Data Type	Mandatory	Description
PATHS	String	Yes	Paths to the local files or OBS files to be uploaded to a DLI group. Multiple paths can be specified at a time.
-k / --kind	String	No	Type of the file to be uploaded, which can be JAR, PyFile, or file.
-g / --group	String	No	Name of the DLI group to which files are to be uploaded.
-tags / --tags	String	No	Tag of the files to be uploaded.
-obs / --obs-bucket	String	No	If the files to be uploaded contain a local path, specify an OBS bucket for transit.
-async / --is-async	Bool	No	Whether to asynchronously upload files. This method is recommended.

Example

- Upload local files to a DLI group.
ma-cli dli-job upload ./test.py -obs \${your-bucket} --kind pyFile

```
(PyTorch-1.8) [ma-user work]$ma-cli dli-job upload ./test.py -obs obs://your-bucket --kind pyFile  
[ OK ] Upload ['test.py'] successfully.
```

- Upload OBS files to a DLI group.
ma-cli dli-job upload obs://your-bucket/test.py --kind pyFile

```
(PyTorch-1.8) [ma-user work]$ma-cli dli-job upload obs://your-bucket/test.py --kind pyFile  
[ OK ] Upload ['test.py'] successfully.
```

Using ma-cli dli-job stop to Stop a DLI Spark Job

Run the **ma-cli dli-job stop** command to stop a DLI Spark job.

```
$ ma-cli dli-job stop -h  
Usage: ma-cli dli-job stop [OPTIONS]
```

Stop DLI spark job by job id.

Example:

Stop training job by job id
ma-cli dli-job stop --job-id \${job_id}

Options:

```
-i, --job-id TEXT      Get DLI spark job event by job id. [required]
-y, --yes              Confirm stop operation.
-D, --config-file TEXT Configure file path for authorization.
-D, --debug            Debug Mode. Shows full stack trace when error occurs.
-P, --profile TEXT     CLI connection profile to use. The default profile is "DEFAULT".
-H, -h, --help         Show this message and exit.
```

Table 5-49 Parameters

Parameter	Data Type	Mandatory	Description
-i / --job-id	String	Yes	DLI Spark job ID
-y / --yes	Bool	No	Whether to forcibly stop a specified DLI Spark job

Example

```
ma-cli dli-job stop -i ${your_job_id}
```

```
(PyTorch-1.8) [ma-user work]$ma-cli dli-job stop -i 4b7c3333-3333-3333-3333-333333333333
[ WARN ] Spark job 4b7c3333-3333-3333-3333-333333333333 will be stopped, do you want to continue (y for confirm)? [y/N]: y
[ OK ] Successfully stop spark batch job [ 4b7c3333-3333-3333-3333-333333333333 ].
```

5.9.8 Using ma-cli to Copy OBS Data

Run the **ma-cli obs-copy [SRC] [DST]** command to copy a local file to an OBS folder or an OBS file or folder to a local path.

```
$ma-cli obs-copy -h
```

Usage: ma-cli obs-copy [OPTIONS] SRC DST

Copy file or directory between OBS and local path. Example:

```
# Upload local file to OBS path
ma-cli obs-copy ./test.zip obs://your-bucket/copy-data/
```

```
# Upload local directory to OBS path
ma-cli obs-copy ./test/ obs://your-bucket/copy-data/
```

```
# Download OBS file to local path
ma-cli obs-copy obs://your-bucket/copy-data/test.zip ./test.zip
```

```
# Download OBS directory to local path
ma-cli obs-copy obs://your-bucket/copy-data/ ./test/
```

Options:

- d, --drop-last-dir Whether to drop last directory when copy folder. if True, the last directory of the source folder will not copy to the destination folder. [default: False]
- C, --config-file PATH Configure file path for authorization.
- D, --debug Debug Mode. Shows full stack trace when error occurs.
- P, --profile TEXT CLI connection profile to use. The default profile is "DEFAULT".
- H, -h, --help Show this message and exit.

Table 5-50 Parameters

Parameter	Type	Mandatory	Description
-d / --drop-last-dir	Bool	No	If you specify this parameter, the last-level directory of the source folder will not be copied to the destination folder. This parameter is valid only for copying folders.

Examples

Upload a file to OBS.

```
$ ma-cli obs-copy ./test.csv obs://${your_bucket}/test-copy/  
[ OK ] local src path: [ /home/ma-user/work/test.csv ]  
[ OK ] obs dst path: [ obs://${your_bucket}/test-copy/ ]
```

Upload a folder to **obs://\${your_bucket}/test-copy/data/**.

```
$ ma-cli obs-copy /home/ma-user/work/data/ obs://${your_bucket}/test-copy/  
[ OK ] local src path: [ /home/ma-user/work/data/ ]  
[ OK ] obs dst path: [ obs://${your_bucket}/test-copy/ ]
```

Upload a folder to **obs://\${your_bucket}/test-copy/** with **--drop-last-dir** specified.

```
$ ma-cli obs-copy /home/ma-user/work/data/ obs://${your_bucket}/test-copy/ --drop-last-dir  
[ OK ] local src path: [ /home/ma-user/work/data ]  
[ OK ] obs dst path: [ obs://${your_bucket}/test-copy/ ]
```

Download a folder from OBS to a local disk.

```
$ ma-cli obs-copy obs://${your_bucket}/test-copy/ ~/work/test-data/  
[ OK ] obs src path: [ obs://${your_bucket}/test-copy/ ]  
[ OK ] local dst path: [ /home/ma-user/work/test-data/ ]
```

5.10 Using MoXing Commands in a Notebook Instance

5.10.1 MoXing Framework Functions

MoXing Framework provides basic common components for MoXing. For example, it facilitates access to Huawei Cloud OBS. Importantly, MoXing Framework is decoupled from specific AI engines and can be seamlessly integrated with all major AI engines (including TensorFlow, MXNet, PyTorch, and MindSpore) supported by ModelArts. MoXing Framework allows you to interact with OBS components using the `mox.file` APIs described in this section.

NOTE

MoXing primarily serves to streamline the process of reading and downloading data from OBS buckets. However, it is not suitable for OBS parallel file systems. You are advised to call the OBS Python SDK to develop production service code. For details, see [API Overview of OBS SDK for Python](#).

Why mox.file

Use Python to open a local file.

```
with open('/tmp/a.txt', 'r') as f:  
    print(f.read())
```

An OBS directory starts with **obs://**, for example, **obs://bucket/XXX.txt**. You cannot directly use the **open** function to open an OBS file. The preceding code for opening a local file will report an error.

With OBS, you can access various tools like SDK, API, OBS console, and OBS Browser. ModelArts mox.file offers a set of APIs that mimic a local file system, enabling easy management of OBS files. For example, you can use the following code to open a file on OBS:

```
import moxing as mox  
with mox.file.File('obs://bucket_name/a.txt', 'r') as f:  
    print(f.read())
```

The following Python code lists a local path:

```
import os  
os.listdir('/tmp/my_dir/')
```

To list an OBS path, add the following code in mox.file:

```
import moxing as mox  
mox.file.list_directory('obs://bucket_name/my_dir/')
```

Importing MoXing Framework

To use the MoXing Framework, first add the MoXing Framework module to the beginning of your code.

Import the MoXing Framework module:

```
import moxing as mox
```

Related Notes

After the MoXing module is introduced, the standard logging module of Python is set to the INFO level, and the version number is printed. You can use the following API to reset the **logging** level:

```
import logging  
  
from moxing.framework.util import runtime  
runtime.reset_logger(level=logging.WARNING)
```

Before introducing MoXing, you can set the **MOX_SILENT_MODE** environment variable to **1** to prevent MoXing from printing the version number. Before importing MoXing, set the environment variables using the following Python code.

```
import os  
os.environ['MOX_SILENT_MODE'] = '1'  
import moxing as mox
```

Data Downloading Acceleration

You can use MoXing Framework to accelerate data downloading for training jobs created using ModelArts preset images. This is suitable when the number of files

ranges from 1 million to 10 million, a single large file, or the file size is greater than 20 GB.

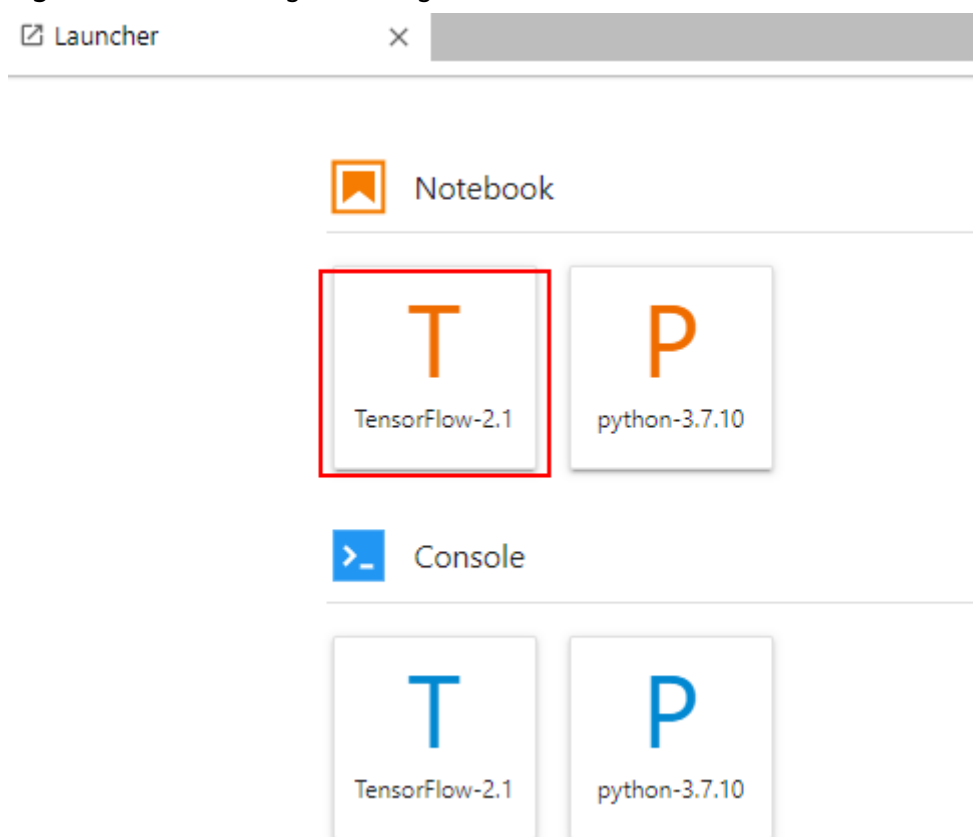
1. Log in to the ModelArts console and choose **Model Training > Training Jobs** in the navigation pane on the left.
2. In the upper right corner of the page, click **Create Training Job**, and set **MA_MOXING_FWVER=2.2.8.0aa484aa** in **Environment Variable** to install the latest MoXing Framework. For details about other parameters, see [Creating a Training Job](#). Then, you can use **moxing.file.copy_parallel** in the training job script to accelerate data downloading.
3. Set **MOX_C_ACCELERATE=0** in **Environment Variable** to disable data downloading acceleration if it is not needed.

5.10.2 Using MoXing in Notebook

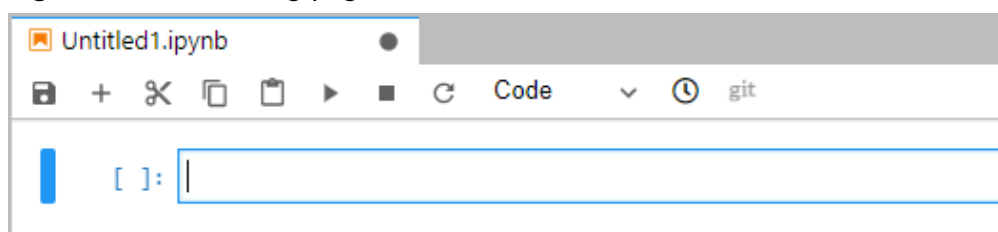
This document describes how to call MoXing Framework APIs in ModelArts.

Logging In to ModelArts and Creating a Notebook Instance

1. Log in to the ModelArts console. In the navigation pane, choose **Development Workspace > Notebook** to access the **Notebook** page.
2. Click **Create**. On the **Create Notebook** page that is displayed, create a notebook instance by referring to [Creating a Notebook Instance \(Default Page\)](#).
3. After a notebook instance is created and enters the **Running** status, click **Open** in the **Operation** column to go to the **JupyterLab Notebook** page.
4. On the **Launcher** page of JupyterLab, for example, click **TensorFlow** to create a file for encoding.

Figure 5-162 Selecting an AI engine

After the file is created, the **JupyterLab** page is displayed by default.

Figure 5-163 Encoding page

Calling `mox.file`.

Enter the following code to implement the following simple functions:

1. Introduce MoXing Framework.
2. Create the **test01** folder in the existing **modelarts-test08/moxing** directory.
3. Check whether the **test01** folder exists. If the folder exists, the preceding operation is successful.

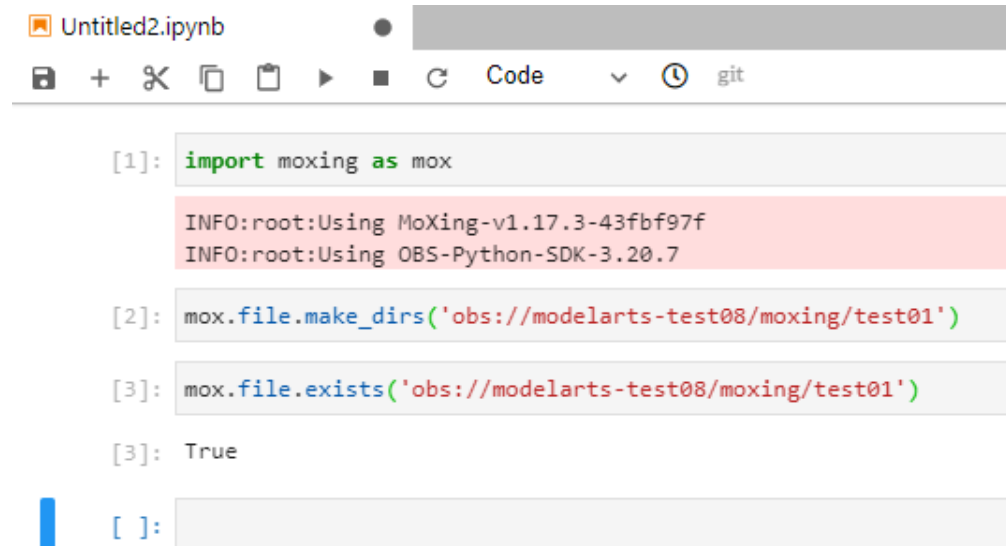
```
import moxing as mox
```

```
mox.file.make_dirs('obs://modelarts-test08/moxing/test01')  
mox.file.exists('obs://modelarts-test08/moxing/test01')
```

Figure 5-164 shows the result. Note that each time you enter a line of code, click **Run**. You can also go to OBS Console and check whether the **test01** folder has

been created in the **modelarts-test08/moxing** directory. For more common MoXing operations, see [Sample Code for Common Operations](#).

Figure 5-164 Example



The screenshot shows a Jupyter Notebook window titled 'Untitled2.ipynb'. The toolbar includes icons for saving, adding, deleting, and running code, along with a 'Code' dropdown and a 'git' icon. The code cell contains the following Python code:

```
[1]: import moxing as mox

INFO:root:Using MoXing-v1.17.3-43fbf97f
INFO:root:Using OBS-Python-SDK-3.20.7

[2]: mox.file.make_dirs('obs://modelarts-test08/moxing/test01')

[3]: mox.file.exists('obs://modelarts-test08/moxing/test01')

[3]: True

[ ]:
```

Copying Data to OBS

On the Notebook JupyterLab page, copy the **yolov8_train_ascend.zip** file to an OBS bucket. The sample code is as follows:

```
import os
import zipfile
import moxing as mox
mox.file.copy('yolov8_train_ascend.zip', 'obs://pcb-data-me/pcb.zip')
```

5.10.3 Introducing MoXing Framework

Before using MoXing Framework, you need to introduce the MoXing Framework module at the beginning of the code.

Introducing MoXing Framework

Run the following code to import the MoXing module:

```
import moxing as mox
```

Related Notes

After the MoXing module is introduced, the standard logging module of Python is set to the INFO level, and the version number is printed. You can use the following API to reset the **logging** level:

```
import logging

from moxing.framework.util import runtime
runtime.reset_logger(level=logging.WARNING)
```

Before introducing MoXing, you can set the **MOX_SILENT_MODE** environment variable to **1** to prevent MoXing from printing the version number. Use the

following Python code to configure the environment variable. You need to configure the environment variables before importing MoXing.

```
import os
os.environ['MOX_SILENT_MODE'] = '1'
import moxing as mox
```

5.10.4 Mapping Between mox.file and Local APIs and Switchover

API Mapping

- Python: local file operation APIs of Python. The APIs can be shifted to the corresponding MoXing file operation APIs (mox.file) by one click.
- mox.file: file operation APIs of MoXing Framework. The APIs correspond to the Python APIs.
- tf.gfile: TensorFlow APIs with the same functions as MoXing file operation APIs. In MoXing, file operation APIs cannot be automatically switched to TensorFlow APIs. The following table lists only the APIs with similar functions.

Table 5-51 API mapping

Python (Local File Operation API)	mox.file (MoXing File Operation API)	tf.gfile (TensorFlow File Operation API)
glob.glob	mox.file.glob	tf.gfile.Glob
os.listdir	mox.file.list_directory(..., recursive=False)	tf.gfile.ListDirectory
os.makedirs	mox.file.make_dirs	tf.gfile.MakeDirs
os.mkdir	mox.file.mk_dir	tf.gfile.MkDir
os.path.exists	mox.file.exists	tf.gfile.Exists
os.path.getsize	mox.file.get_size	-
os.path.isdir	mox.file.is_directory	tf.gfile.IsDirectory
os.remove	mox.file.remove(..., recursive=False)	tf.gfile.Remove
os.rename	mox.file.rename	tf.gfile.Rename
os.scandir	mox.file.scan_dir	-
os.stat	mox.file.stat	tf.gfile.Stat
os.walk	mox.file.walk	tf.gfile.Walk
open	mox.file.File	tf.gfile.FastGFile(tf.gfile.Gfile)
shutil.copyfile	mox.file.copy	tf.gfile.Copy
shutil.copytree	mox.file.copy_parallel	-

Python (Local File Operation API)	mox.file (MoXing File Operation API)	tf.gfile (TensorFlow File Operation API)
shutil.rmtree	mox.file.remove(..., recursive=True)	tf.gfile.DeleteRecursively

5.10.5 Sample Code for Common Operations

Data Reads and Writes

- Read an OBS file.

For example, if you read the **obs://bucket_name/obs_file.txt** file, the content is returned as strings.

```
import moxing as mox
file_str = mox.file.read('obs://bucket_name/obs_file.txt')
```

You can also open the file object and read data from it. Both methods are the same.

```
import moxing as mox
with mox.file.File('obs://bucket_name/obs_file.txt', 'r') as f:
    file_str = f.read()
```

- Read a line from a file. A string that ends with a newline character is returned. You can also open the file object in OBS.

```
import moxing as mox
with mox.file.File('obs://bucket_name/obs_file.txt', 'r') as f:
    file_line = f.readline()
```

- Read all lines from a file. A list is returned, in which each element is a line and ends with a newline character.

```
import moxing as mox
with mox.file.File('obs://bucket_name/obs_file.txt', 'r') as f:
    file_line_list = f.readlines()
```

- Read an OBS file in binary mode.

For example, if you read the **obs://bucket_name/obs_file.bin** file, the content is returned as bytes.

```
import moxing as mox
file_bytes = mox.file.read('obs://bucket_name/obs_file.bin', binary=True)
```

You can also open the file object and read data from it. Both methods are the same.

```
import moxing as mox
with mox.file.File('obs://bucket_name/obs_file.bin', 'rb') as f:
    file_bytes = f.read()
```

One or all lines in a file opened in binary mode can be read with the same method.

- Write a string to a file.

For example, write **Hello World!** into the **obs://bucket_name/obs_file.txt** file.

```
import moxing as mox
mox.file.write('obs://bucket_name/obs_file.txt', 'Hello World!')
```

You can also open the file object and write data into it. Both methods are the same.

```
import moxing as mox
with mox.file.File('obs://bucket_name/obs_file.txt', 'w') as f:
    f.write('Hello World!')
```

 NOTE

When you open a file in write mode or call **mox.file.write**, if the file to be written does not exist, the file will be created. If the file to be written already exists, the file is overwritten.

- Append content to an OBS file.

For example, append **Hello World!** to the **obs://bucket_name/obs_file.txt** file.

```
import moxing as mox
mox.file.append('obs://bucket_name/obs_file.txt', 'Hello World!')
```

You can also open the file object and append content to it. Both methods are the same.

```
import moxing as mox
with mox.file.File('obs://bucket_name/obs_file.txt', 'a') as f:
    f.write('Hello World!')
```

When you open a file in append mode or call **mox.file.append**, if the file to be appended does not exist, the file will be created. If the file to be appended already exists, the content is directly appended.

If the size of the source file to be appended is large, for example, the **obs://bucket_name/obs_file.txt** file exceeds 5 MB, the append performance is low.

 NOTE

If the file object is opened in write or append mode, when the **write** function is called, the content to be written is temporarily stored in the cache until the file object is closed (the file object is automatically closed when the **with** statement exits). Alternatively, you can call the **close()** or **flush()** function of the file object to write the file content.

List

- List an OBS directory. Only the top-level result (relative path) is returned. Recursive listing is not performed.

For example, if you list **obs://bucket_name/object_dir**, all files and folders in the directory are returned, but recursive queries are not performed.

Assume that **obs://bucket_name/object_dir** is in the following structure:

```
bucket_name
|- object_dir
   |- dir0
   |- file00
   |- file1
```

Call the following code:

```
import moxing as mox
mox.file.list_directory('obs://bucket_name/object_dir')
```

The following list is returned:

```
['dir0', 'file1']
```

- Recursively list an OBS directory. All files and folders (relative paths) in the directory are returned, and recursive queries are performed.

Assume that **obs://bucket_name/object_dir** is in the following structure:

```
bucket_name
|- object_dir
   |- dir0
   |- file00
   |- file1
```

Call the following code:

```
import moxing as mox
mox.file.list_directory('obs://bucket_name/object_dir', recursive=True)
```

The following list is returned:

```
['dir0', 'dir0/file00', 'file1']
```

Create a Folder

Create an OBS directory, that is, an OBS folder. Recursive creation is supported. That is, if the **sub_dir_0** folder does not exist, it is automatically created. If the **sub_dir_0** folder exists, no folder will be created.

```
import moxing as mox
mox.file.make_dirs('obs://bucket_name/sub_dir_0/sub_dir_1')
```

Query

- Check whether an OBS file exists. If the file exists, **True** is returned. If the file does not exist, **False** is returned.

```
import moxing as mox
mox.file.exists('obs://bucket_name/sub_dir_0/file.txt')
```

- Check whether an OBS folder exists. If the folder exists, **True** is returned. If the folder does not exist, **False** is returned.

```
import moxing as mox
mox.file.exists('obs://bucket_name/sub_dir_0/sub_dir_1')
```

NOTE

OBS allows files and folders with the same name exist (not allowed in UNIX). If a file or folder with the same name exists, for example, **obs://bucket_name/sub_dir_0/abc**, when **mox.file.exists** is called, **True** is returned regardless of whether **abc** is a file or folder.

- Check whether an OBS path is a folder. If it is a folder, **True** is returned. If it is not a folder, **False** is returned.

```
import moxing as mox
mox.file.is_directory('obs://bucket_name/sub_dir_0/sub_dir_1')
```

NOTE

OBS allows files and folders with the same name exist (not allowed in UNIX). If a file or folder with the same name exists, for example, **obs://bucket_name/sub_dir_0/abc**, when **mox.file.is_directory** is called, **True** is returned.

- Obtain the size of an OBS file, in bytes.

For example, obtain the size of **obs://bucket_name/obs_file.txt**.

```
import moxing as mox
mox.file.get_size('obs://bucket_name/obs_file.txt')
```

- Recursively obtain the size of all files in an OBS folder, in bytes.

For example, obtain the total size of all files in the **obs://bucket_name/object_dir** directory.

```
import moxing as mox
mox.file.get_size('obs://bucket_name/object_dir', recursive=True)
```

- Obtain the **stat** information about an OBS file or folder. The **stat** information contains the following:

- **length**: file size
- **mtime_nsec**: creation timestamp

- **is_directory**: whether the path is a folder

For example, if you want to query the OBS file **obs://bucket_name/obs_file.txt**, you can replace the file path with a folder path.

```
import moxing as mox
stat = mox.file.stat('obs://bucket_name/obs_file.txt')
print(stat.length)
print(stat.mtime_nsec)
print(stat.is_directory)
```

Delete

- Delete an OBS file.

For example, delete **obs://bucket_name/obs_file.txt**.

```
import moxing as mox
mox.file.remove('obs://bucket_name/obs_file.txt')
```

- Delete an OBS folder and recursively delete all content in the folder. If the folder does not exist, an error is reported.

For example, delete all content in **obs://bucket_name/sub_dir_0**.

```
import moxing as mox
mox.file.remove('obs://bucket_name/sub_dir_0', recursive=True)
```

Move and Copy

- Move an OBS file or folder. The move operation is implemented by copying and deleting data.

- Move an OBS file to another OBS file. For example, move **obs://bucket_name/obs_file.txt** to **obs://bucket_name/obs_file_2.txt**.

```
import moxing as mox
mox.file.rename('obs://bucket_name/obs_file.txt', 'obs://bucket_name/obs_file_2.txt')
```

NOTE

The move and copy operation must be performed in the same bucket.

- Move an OBS file to a local file. For example, move **obs://bucket_name/obs_file.txt** to **/tmp/obs_file.txt**.

```
import moxing as mox
mox.file.rename('obs://bucket_name/obs_file.txt', '/tmp/obs_file.txt')
```

- Move a local file to an OBS file. For example, move **/tmp/obs_file.txt** to **obs://bucket_name/obs_file.txt**.

```
import moxing as mox
mox.file.rename('/tmp/obs_file.txt', 'obs://bucket_name/obs_file.txt')
```

- Move a local file to another local file. For example, move **/tmp/obs_file.txt** to **/tmp/obs_file_2.txt**. This operation is equivalent to **os.rename**.

```
import moxing as mox
mox.file.rename('/tmp/obs_file.txt', '/tmp/obs_file_2.txt')
```

You can move folders in the same way. If you move a folder, all content in the folder is moved recursively.

- Copy a file. **mox.file.copy** can be used to perform operations only on files. To perform operations on folders, use **mox.file.copy_parallel**.

- Copy an OBS file to another OBS path. For example, copy **obs://bucket_name/obs_file.txt** to **obs://bucket_name/obs_file_2.txt**.

```
import moxing as mox
mox.file.copy('obs://bucket_name/obs_file.txt', 'obs://bucket_name/obs_file_2.txt')
```

- Copy an OBS file to a local path, that is, download an OBS file. For example, download **obs://bucket_name/obs_file.txt** to **/tmp/obs_file.txt**.

```
import moxing as mox
mox.file.copy('obs://bucket_name/obs_file.txt', '/tmp/obs_file.txt')
```
- Copy a local file to OBS, that is, upload an OBS file. For example, upload **/tmp/obs_file.txt** to **obs://bucket_name/obs_file.txt**.

```
import moxing as mox
mox.file.copy('/tmp/obs_file.txt', 'obs://bucket_name/obs_file.txt')
```
- Copy a local file to another local path. This operation is equivalent to **shutil.copyfile**. For example, copy **/tmp/obs_file.txt** to **/tmp/obs_file_2.txt**.

```
import moxing as mox
mox.file.copy('/tmp/obs_file.txt', '/tmp/obs_file_2.txt')
```
- Copy a folder. **mox.file.copy_parallel** can be used to perform operations only on folders. To perform operations on files, use **mox.file.copy**.
 - Copy an OBS file to another OBS path. For example, copy **obs://bucket_name/sub_dir_0** to **obs://bucket_name/sub_dir_1**.

```
import moxing as mox
mox.file.copy_parallel('obs://bucket_name/sub_dir_0', 'obs://bucket_name/sub_dir_1')
```
 - Copy an OBS folder to a local path, that is, download an OBS folder. For example, download **obs://bucket_name/sub_dir_0** to **/tmp/sub_dir_0**.

```
import moxing as mox
mox.file.copy_parallel('obs://bucket_name/sub_dir_0', '/tmp/sub_dir_0')
```
 - Copy a local folder to OBS, that is, upload an OBS folder. For example, upload **/tmp/sub_dir_0** to **obs://bucket_name/sub_dir_0**.

```
import moxing as mox
mox.file.copy_parallel('/tmp/sub_dir_0', 'obs://bucket_name/sub_dir_0')
```
 - Copy a local folder to another local path. This operation is equivalent to **shutil.copytree**. For example, copy **/tmp/sub_dir_0** to **/tmp/sub_dir_1**.

```
import moxing as mox
mox.file.copy_parallel('/tmp/sub_dir_0', '/tmp/sub_dir_1')
```

5.10.6 Sample Code for Advanced MoXing Usage

If you are familiar with common operations, the MoXing Framework API document, and common Python code, you can refer to this section to use advanced MoXing Framework functions.

Closing a File After File Reading Is Completed

When you read an OBS file, you are establishing an HTTP connection to access the network stream. Once done, close the file immediately. To prevent you from forgetting to close a file, you are advised to use the **with** statement. When the **with** statement exits, the **close()** function of the **mox.file.File** object is automatically called.

```
import moxing as mox
with mox.file.File('obs://bucket_name/obs_file.txt', 'r') as f:
    data = f.readlines()
```

Reading or Writing an OBS File Using pandas

- Use **pandas** to read an OBS file.

```
import pandas as pd
import moxing as mox
```



```
with mox.file.File("obs://bucket_name/b.txt", "r") as f:  
    csv = pd.read_csv(f)
```

- Use **pandas** to write an OBS file.

```
import pandas as pd  
import moxing as mox  
df = pd.DataFrame({'col1': [1, 2], 'col2': [3, 4]})  
with mox.file.File("obs://bucket_name/b.txt", "w") as f:  
    df.to_csv(f)
```

Reading an Image Using a File Object

When OpenCV is used to open an image, the OBS path cannot be passed and the image must be read using a file object. The following code cannot read the image:

```
import cv2  
cv2.imread('obs://bucket_name/xxx.jpg', cv2.IMREAD_COLOR)
```

Modify the code as follows:

```
import cv2  
import numpy as np  
import moxing as mox  
img = cv2.imdecode(np.fromstring(mox.file.read('obs://bucket_name/xxx.jpg', binary=True), np.uint8),  
cv2.IMREAD_COLOR)
```

Reconstructing an API That Does Not Support OBS Paths to One That Does

In **pandas**, **to_hdf** and **read_hdf** used to read and write H5 files do not support OBS paths, nor do they support file objects to be entered. The following code may cause errors:

```
import pandas as pd  
df = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6]}, index=['a', 'b', 'c'])  
df.to_hdf('obs://wolfros-net/hdftest.h5', key='df', mode='w')  
pd.read_hdf('obs://wolfros-net/hdftest.h5')
```

The API compiled using the pandas source code is rewritten to support OBS paths.

- Write H5 to OBS = Write H5 to the local cache + Upload the local cache to OBS + Delete the local cache
- Read H5 from OBS = Download H5 to the local cache + Read the local cache + Delete the local cache

That is, write the following code at the beginning of the script to enable **to_hdf** and **read_hdf** to support OBS paths:

```
import os  
import moxing as mox  
import pandas as pd  
from pandas.io import pytables  
from pandas.core.generic import NDFrame  
  
to_hdf_origin = getattr(NDFrame, 'to_hdf')  
read_hdf_origin = getattr(pytables, 'read_hdf')  
  
def to_hdf_override(self, path_or_buf, key, **kwargs):  
    tmp_dir = '/cache/hdf_tmp'  
    file_name = os.path.basename(path_or_buf)  
    mox.file.make_dirs(tmp_dir)  
    local_file = os.path.join(tmp_dir, file_name)  
    to_hdf_origin(self, local_file, key, **kwargs)  
    mox.file.copy(local_file, path_or_buf)  
    mox.file.remove(local_file)
```

```
def read_hdf_override(path_or_buf, key=None, mode='r', **kwargs):
    tmp_dir = '/cache/hdf_tmp'
    file_name = os.path.basename(path_or_buf)
    mox.file.make_dirs(tmp_dir)
    local_file = os.path.join(tmp_dir, file_name)
    mox.file.copy(path_or_buf, local_file)
    result = read_hdf_origin(local_file, key, mode, **kwargs)
    mox.file.remove(local_file)
    return result

setattr(NDFrame, 'to_hdf', to_hdf_override)
setattr(pytables, 'read_hdf', read_hdf_override)
setattr(pd, 'read_hdf', read_hdf_override)
```

Use MoXing to Enable h5py.File to Support OBS

```
import os
import h5py
import numpy as np
import moxing as mox

h5py_File_class = h5py.File

class OBSFile(h5py_File_class):
    def __init__(self, name, *args, **kwargs):
        self._tmp_name = None
        self._target_name = name
        if name.startswith('obs://'):
            self._tmp_name = name.replace('/', '_')
            if mox.file.exists(name):
                mox.file.copy(name, os.path.join('cache', 'h5py_tmp', self._tmp_name))
            name = self._tmp_name

        super(OBSFile, self).__init__(name, *args, **kwargs)

    def close(self):
        if self._tmp_name:
            mox.file.copy(self._tmp_name, self._target_name)

        super(OBSFile, self).close()

setattr(h5py, 'File', OBSFile)

arr = np.random.randn(1000)
with h5py.File('obs://bucket/random.hdf5', 'r') as f:
    f.create_dataset("default", data=arr)

with h5py.File('obs://bucket/random.hdf5', 'r') as f:
    print(f.require_dataset("default", dtype=np.float32, shape=(1000,)))
```

6 Preparing and Processing Data

6.1 Preparing Data

ModelArts is a one-stop AI development platform that supports model lifecycle development, including data processing, algorithm development, model training, and model deployment. In addition, ModelArts provides AI Gallery that can be used to share data, algorithms, and models. ModelArts data management provides end-to-end data preparation, processing, and labeling.

ModelArts data management provides the following functions for you to obtain high-quality AI data:

- Data acquisition
 - Allows you to import data from OBS and MRS.
 - Provides 18+ data augmentation operators to increase data volume for training.
- Improved data quality
 - Allows you to preview various formats of data including images, text, audios, and videos, helping you identify data quality.
 - Allows you to filter data by multiple search criteria, such as sample attributes and labeling information.
 - Provides 12+ labeling tools for refined, scenario-based, and professional data labeling.
 - Performs feature analysis based on samples and labeling results, helping you understand data quality.
- More efficient data preparation
 - Allows you to manage data by version for more efficient data management.
 - Provides capabilities such as interactive labeling and auto labeling for more efficient data labeling.
 - Enables team labeling and team labeling management for labeling a large amount of data.

6.2 Creating a ModelArts Dataset

Before using ModelArts to prepare data, create a dataset. Then, you can perform operations on the dataset, such as importing data, analyzing data, and labeling data.

NOTE

Datasets are supported only in the following regions: CN North-Beijing⁴, CN Southwest-Guiyang¹, CN-Hong Kong, AP-Singapore, AP-Bangkok, AP-Jakarta, AF-Johannesburg, LA-Santiago, LA-Sao Paulo¹, and LA-Mexico City².

Dataset Types

ModelArts supports the following types of datasets:

- Images: in .jpg, .png, .jpeg, or .bmp format for image classification, image segmentation, and object detection
- Audio: in .wav format for sound classification, speech labeling, and speech paragraph labeling
- Text: in .txt or .csv format for text classification, named entity recognition, and text triplet labeling
- Video: in .mp4 format for video labeling
- Free format: allows data in any format. Labeling is not available for free format data. The free format applies if labeling is not required or needs to be customized. If your dataset needs to contain data in multiple formats or your data format does not meet the requirements of other types of datasets, you can select a dataset in free format.
- Table
Table: applies to structured data processing such as tables. The file format can be CSV. Tables cannot be labeled but you can preview up to 100 data records in a table.

Dataset Functions

Different types of datasets support different functions, such as auto labeling and team labeling. For details, see [Table 6-1](#).

Table 6-1 Functions supported by different types of datasets

Dataset Type	Label Dataset	Classification Dataset	Image Dataset	Video Dataset	Audio Dataset	Text Dataset	Time Series Dataset	Auto Grouping	Data Feature Engineering
Image Dataset	Supported	Supported	Supported	Supported	Supported	Supported	Supported	Supported	Supported

D a t a s e t T y p e	L a b e l T y p e	C l a s s T y p e	I m p u t T y p e	E x p o s t T y p e	P r o b l e m T y p e	M a t r i x T y p e	M a t r i x T y p e	A u t o g r a p h T y p e	T e m p l a t e T y p e	Auto Grouping	Data Feature Engineering
O b j e c t d e t e c t i o n	S u p p o r t e d	S u p p o r t e d	S u p p o r t e d	S u p p o r t e d	S u p p o r t e d	S u p p o r t e d	S u p p o r t e d	S u p p o r t e d	S u p p o r t e d	Supported	Supported

[illegible]

[illegible]

D a t a s e t T y p e	L a b e l T y p e	C l a s s T y p e	I m p u t T y p e	E x p o s t T y p e	P r o c e s s i n g T y p e	M a i n f r a s t r u c t u r e	M a i n f r a s t r u c t u r e	A u t o g r o u p i n g	T e m p l a t e	Auto Grouping	Data Feature Engineering
S u p p l e m e n t s	S u p p l e m e n t s	S u p p l e m e n t s	-	S u p p l e m e n t s	S u p p l e m e n t s	S u p p l e m e n t s	-	-	-		-

D a t a s e t T y p e	L a b e l T y p e	C l a s s T y p e	I m p l e m e n t T y p e	E n g i n e e r i n g T y p e	P r o c e s s i n g T y p e	M a c h i n e l e m e n t T y p e	M a c h i n e l e m e n t T y p e	A u t o m a t i c a l l y n g T y p e	T e m p l a t e T y p e	Auto Grouping	Data Feature Engineering
S u p p l e m e n t T y p e	S u p p l e m e n t T y p e	S u p p l e m e n t T y p e	-	S u p p l e m e n t T y p e	S u p p l e m e n t T y p e	S u p p l e m e n t T y p e	-	S u p p l e m e n t T y p e	-		-

D a t a s e t T y p e	L a b e l T y p e	C a t e g o r y T y p e	I m p u t T y p e	E x p o u r e T y p e	P r e p r o c e s s i n g T y p e	M a i n t e n a n c e T y p e	M a i n t e n a n c e T y p e	A u t o g r o u p i n g T y p e	T e x t T y p e	Auto Grouping	Data Feature Engineering
T e x t	T e x t	S u p p o r t e d	S u p p o r t e d	-	S u p p o r t e d	S u p p o r t e d	S u p p o r t e d	-	S u p p o r t e d	-	-

D a t a s e t T y p e	L a b e l i n g T y p e	C l a s s i f i c a t i o n	I m p u t D a t a S e t	E x p l o r a t o r y D a t a S e t	P r e p r o c e s s i n g D a t a S e t	M a i n f i l e S y s t e m D a t a S e t	M a i n f i l e S y s t e m D a t a S e t	A u t o g r o u p i n g D a t a S e t	T r a n s f e r r i n g D a t a S e t	Auto Grouping	Data Feature Engineering
N a m e d e n t i t y r e c o g n i t i o n	S u p p o r t e d	S u p p o r t e d	-	S u p p o r t e d	S u p p o r t e d	S u p p o r t e d	-	S u p p o r t e d	-		-

D a t a S e l e c t T y p e	L a b e l i n g T y p e	C r e a t i n g D a t a S e t	I m p o r t i n g D a t a	E x p o r t i n g D a t a	P u b l i s h i n g D a t a S e t	M o d i f y i n g D a t a S e t	M a n a g i n g D a t a S e t V e r s i o n s	A u t o l a b e l i n g	T e a m	Auto Grouping	Data Feature Engineering
	T e x t t r i p l e t	S u p p o r t e d	S u p p o r t e d	-	S u p p o r t e d	S u p p o r t e d	S u p p o r t e d	-	S u p p o r t e d	-	-
V i d e o	V i d e o	S u p p o r t e d	S u p p o r t e d	-	S u p p o r t e d	S u p p o r t e d	S u p p o r t e d	-	-	-	-

D a t a s e t T y p e	L a b e l T y p e	C l a s s T y p e	I m a g e T y p e	E m b e d d i n g T y p e	P r o b a b i l i t y T y p e	M u l t i m e d i a T y p e	M u l t i m e d i a T y p e	A u d i o T y p e	T a b l e T y p e	Auto Grouping	Data Feature Engineering
F u r t h e r m o r e a t t r i b u t e s	F u r t h e r m o r e a t t r i b u t e s	S u p p l e m e n t a r y	-	-	S u p p l e m e n t a r y	S u p p l e m e n t a r y	S u p p l e m e n t a r y	-	-	-	-
T a b l e	T a b l e	S u p p l e m e n t a r y	S u p p l e m e n t a r y	-	S u p p l e m e n t a r y	S u p p l e m e n t a r y	S u p p l e m e n t a r y	-	-	-	-

Specifications Restrictions

- The maximum numbers of samples and labels in a single text, video, or audio database other than a table dataset are 1,000,000 and 10,000, respectively.
- The maximum size of a sample in a single text, video, or audio database other than an image dataset is 5 GB.

- The maximum size of an image for object detection, image segmentation, or image classification is 25 MB.
- The manifest file cannot be larger than 5 GB.
- The text file in a line cannot be larger than 100 KB.
- The dataset labeling result file cannot be larger than 100 MB.

Prerequisites

- You have been authorized to access OBS. To do so, go to the ModelArts management console. In the navigation pane on the left, choose **Permission Management**, and add access authorization using an agency.
- OBS buckets and folders for storing data are available. In addition, the OBS buckets and ModelArts are in the same region. OBS parallel file systems are not supported. Select object storage.
- ModelArts does not support encrypted OBS buckets. When creating an OBS bucket, do not enable bucket encryption.

Creating a Dataset (Image, Audio, Text, Video, and Free Format)

1. Log in to the [ModelArts management console](#). In the navigation pane on the left, choose **Asset Management > Datasets**.
2. Click **Create**. On the **Create Dataset** page, create a dataset based on the data type and data labeling requirements. Enter the basic information about the dataset.

Figure 6-1 Parameters

★ Data Type: Images, Audio, Text, Video, Free format, Table. Supported formats: .jpg, .png, .jpeg, .bmp

★ Data Source: OBS, Local file

★ Import Mode: Path. You can save the dataset file to be imported to the OBS path that you have permission to access. [Labeling file format](#)

★ Import Path: Select an OBS path. You can import up to 1,000,000 samples and 1,000 labels.

★ Labeling Status: Unlabeled, Labeled

★ Output Dataset Path: Select an OBS path. Path for storing output files such as labeled files. The path cannot be the same as the import path or subdirectory of the import path.

- **Name:** Enter a custom dataset name.
- **Description:** Enter the details about the dataset
- **Data Type:** Select a data type based on your needs.
- **Data Source**
 - i. Importing data from OBS
If you have prepared data on OBS, set **Data Source** to **OBS** and configure **Import Path**, **Labeling Status**, and **Output Dataset Path**. If **Labeling Status** is set to **Labeled**, you need to also configure

Labeling Format. The labeling formats of the input data vary depending on the dataset type. For details about the labeling formats supported by ModelArts, see [Dataset Functions](#).

ii. Importing data from a local path

ModelArts also allows you to upload data from a local path. To do so, set **Data Source** to **Local file**, upload data, and configure **Labeling Status** and **Output Dataset Path**. Click **Upload data** to select the local file for uploading. Select a labeling format when the labeling status is **Labeled**. The labeling formats of the input data vary depending on the dataset type. For details about the labeling formats supported by ModelArts, see [Dataset Functions](#).

Figure 6-2 Selecting **Local file**

The screenshot displays the configuration interface for selecting a local file as the data source. It includes the following elements:

- Data Source:** Two buttons, 'OBS' and 'Local file'. The 'Local file' button is highlighted in blue.
- Storage Path:** A text input field with the placeholder 'Select an OBS path.' and a file selection icon. Below it, a note states: 'You can import up to 1,000,000 samples and 1,000 labels.'
- Uploading Data:** A button labeled 'Upload data' with a plus icon.
- Labeling Status:** Two buttons, 'Unlabeled' and 'Labeled'. The 'Unlabeled' button is highlighted in blue.
- Output Dataset Path:** A text input field with the placeholder 'Select an OBS path.' and a file selection icon.

- For details about parameters, see [Table 6-2](#).

Table 6-2 Dataset parameters

Parameter	Description
Import Path	OBS path from which your data is to be imported. This path is used as the data storage path of the dataset. NOTE OBS parallel file systems are not supported. Select an OBS bucket. When you create a dataset, data in the OBS path will be imported to the dataset. If you modify data in OBS, the data in the dataset will be inconsistent with that in OBS. As a result, certain data may be unavailable. If you need to modify data in a dataset, see Import Mode or Importing Data from an OBS Path to ModelArts. If the numbers of samples and labels of the dataset exceed quotas, importing the samples and labels will fail.

Parameter	Description
Labeling Status	Labeling status of the selected data, which can be Unlabeled or Labeled. If you select Labeled, specify a labeling format and ensure the data file complies with format specifications. Otherwise, the import may fail. Only image (object detection, image classification, and image segmentation), audio (sound classification), and text (text classification) labeling tasks support the import of labeled data.
Output Dataset Path	OBS path where your labeled data is stored. NOTE • Ensure that your OBS path name contains letters, digits, and underscores (_) and does not contain special characters, such as ~'@#\$\$%^&*{}[];+=<>/ and spaces. • The dataset output path cannot be the same as the data input path or subdirectory of the data input path. • It is a good practice to select an empty directory as the dataset output path. • OBS parallel file systems are not supported. Select an OBS bucket.
Advanced Feature Settings - Import by Tag	This function is disabled by default. You can enable it to import resources by tag. Import by Tag enables the system to automatically obtain the labels of the current dataset. Click Add Label to add a label. This field is optional. After importing the data, you can add or delete labels during data labeling.

3. After setting the parameters, click **Submit**.

Creating a Dataset (Table)

1. Log in to the [ModelArts management console](#). In the navigation pane on the left, choose **Asset Management > Datasets**.
2. Click **Create**. On the **Create Dataset** page, create a dataset based on the data type and data labeling requirements. Enter the basic information about the dataset.

Figure 6-3 Parameters of a table dataset

* Data Type

Images

Audio

Text

Video

Free format

Table

Data Source

OBS

DWS

DLI

MRS

Local file

File Path

Select an OBS path.

Contain Table Header

☒

* Schema ⓘ

Column Name

Type

String

⌵

⊕ Add Schema

* Output Dataset Path

Select an OBS path.

- **Name:** Enter a custom dataset name.
- **Description:** Enter the details about the dataset
- **Data Type:** Select a data type based on your needs.
- For more details about parameters, see [Table 6-3](#).

Table 6-3 Dataset parameters

Parameter	Description
Data Source (OBS)	• File Path: Browse all OBS buckets of the account and select the directory where the data file to be imported is located. • Contain Table Header: This setting is enabled by default, indicating that the imported file contains table headers. - If the original table contains table headers and this setting is enabled, first rows (table header) of the imported file are used as column names. You do not need to modify the schema information. - If the original table does not contain table headers, you need to disable this setting and change column names in Schema to attr_1, attr_2, ..., and attr_n. attr_n is the last column, indicating the prediction column. For details about OBS functions, see Object Storage Service Console Operation Guide.

Parameter	Description
Data Source (MRS)	● Cluster Name: All MRS clusters of the current account are automatically displayed. However, streaming clusters do not support data import. Select the required cluster from the drop-down list. ● File Path: Enter the HDFS file path based on the selected cluster. ● Contain Table Header: If this setting is enabled, the imported file contains table headers. For details about MRS functions, see MapReduce Service User Guide.
Local file	Storage Path: Select an OBS path.
Schema	Names and types of table columns, which must be the same as those of the imported data. Set the column name based on the imported data and select the column type. For details about the supported types, see Table 6-4. Click Add Schema to add a new record. When creating a dataset, you must specify a schema. Once created, the schema cannot be modified. When data is imported from OBS, the schema of the CSV file in the file path is automatically obtained. If the schemas of multiple CSV files are inconsistent, an error will be reported. NOTE After you select data from OBS, column names in Schema are automatically displayed, which is the first-row data of the table by default. To ensure the correct prediction code, you need to change column names in Schema to attr_1, attr_2, ..., and attr_n. attr_n is the last column, indicating the prediction column.
Output Dataset Path	OBS path for storing table data. The data imported from the data source is stored in this path. The path cannot be the same as the file path in the OBS data source or subdirectories of the file path. After a table dataset is created, the following four directories are automatically generated in the storage path: ● annotation: version publishing directory. Each time a version is published, a subdirectory with the same name as the version is generated in this directory. ● data: data storage directory. Imported data is stored in this directory. ● logs: directory for storing logs. ● temp: temporary working directory.

Table 6-4 Schema data types

Type	Description	Storage Space	Range
String	String type	-	-
Short	Signed integer	2 bytes	-32768-32767
Int	Signed integer	4 bytes	-2147483648 to 2147483647
Long	Signed integer	8 bytes	-9223372036854775808 to 9223372036854775807
Double	Double-precision floating point	8 bytes	-
Float	Single-precision floating point	4 bytes	-
Byte	Signed integer	1 byte	-128-127
Date	Date type in the format of "yyyy-MM-dd", for example, 2014-05-29	-	-
Timestamp	Timestamp that represents date and time in the format of "yyyy-MM-dd HH:mm:ss"	-	-
Boolean	Boolean type	1 byte	TRUE/FALSE

 NOTE

When using a CSV file, pay attention to the following:

- When the data type is set to **String**, the data in the double quotation marks is regarded as one record by default. Ensure the double quotation marks in the same row are closed. Otherwise, the data will be too large to display.
- If the number of columns in a row of the CSV file is different from that defined in the schema, the row will be ignored.

3. After setting the parameters, click **Submit**.

Modifying the Basic Information of a Dataset

1. Log in to the [ModelArts management console](#). In the navigation pane on the left, choose **Asset Management > Datasets**.
2. In the dataset list, locate the target dataset, and choose **More > Modify** in the **Operation** column. Modify the basic information and click **OK**.

Table 6-5 Parameters

Parameter	Description
Name	Name of a dataset, which must contain 1 to 64 characters long and start with a letter. Only letters, digits, underscores (_), and hyphens (-) are allowed. The name must start with a letter.
Description	Brief description of the dataset.

6.3 Importing Data to a ModelArts Dataset

6.3.1 Introduction to Data Importing

After a dataset is created, you can import more data. ModelArts allows you to import data from different data sources.

- [Importing Data from OBS](#)
- [Importing Data from Local Files](#)

File Data Sources

You can import data from OBS or a local file. After the import, the data from the import path is automatically synchronized to the data source path of the dataset.

- **OBS:** Import data from an OBS path or a manifest file.
- **Local file:** Import local data that has been uploaded to an OBS path.

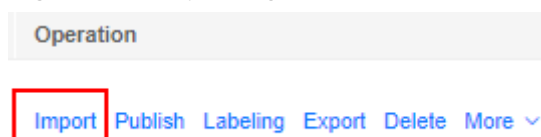
Table Data Sources

You can import data from OBS, MRS, and local files.

Import Mode

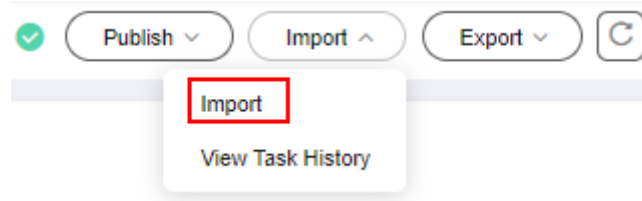
There are five modes for importing data to a dataset.

- When you create a dataset, select an import path. The data is automatically synchronized from the import path.
- After a dataset is created, click **Import** in the **Operation** column on the dataset list page.

Figure 6-4 Importing data on the dataset list page

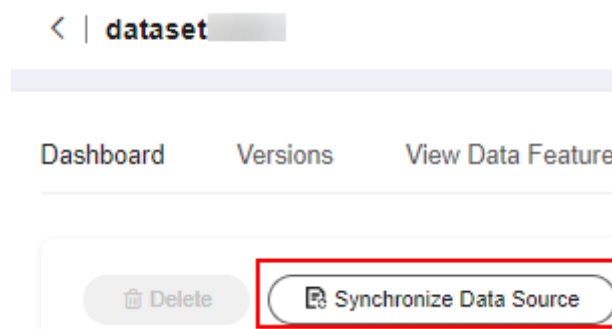
- On the dataset list page, click a dataset. On the dataset details page, choose **Import > Import**.

Figure 6-5 Importing data on the dataset details page



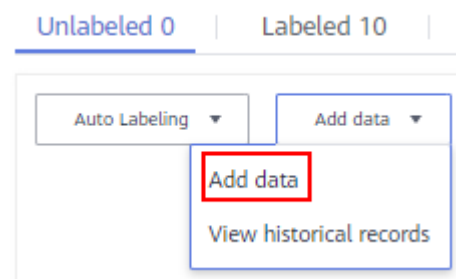
- On the dataset list page, click a dataset. On the dataset details page, click **Synchronize Data Source** to synchronize data from OBS.

Figure 6-6 Synchronizing data sources on the dataset details page



- Add data on the labeling job details page.

Figure 6-7 Adding data on the labeling job details page



6.3.2 Importing Data from OBS

6.3.2.1 Introduction to Importing Data from OBS

Import Modes

You can import data from OBS through an OBS path or a manifest file.

- OBS path:** indicates that the dataset to be imported has been stored in an OBS path. In this case, select an OBS path that you can access. In addition, the directory structure in the OBS path must comply with the specifications. For details, see [Specifications for Importing Data from an OBS Directory](#). This import mode is available only for the following types of datasets: **Image classification**, **Object detection**, **Text classification**, **Table**, and **Sound classification**. For other types of datasets, data can be imported only through a manifest file.

- Manifest file: indicates that the dataset file is in the manifest format and the manifest file has been uploaded to OBS. The manifest file defines the mapping between labeling objects and content. For details about the specifications of manifest files, see [Specifications for Importing a Manifest File](#).

 NOTE

Before importing an object detection dataset, ensure that the labeling range of the labeling file does not exceed the size of the original image. Otherwise, the import may fail.

Table 6-6 Import modes supported by datasets

Dat aset Typ e	Labeling Type	From an OBS Path	From a Manifest File
Ima ges	Image classificati on	Supported You can import unlabeled or labeled data. Format specifications of labeled data: Image Classification	Supported You can import unlabeled or labeled data. Format specifications of labeled data: Image Classification
	Object detection	Supported You can import unlabeled or labeled data. Format specifications of labeled data: Object Detection	Supported You can import unlabeled or labeled data. Format specifications of labeled data: Object Detection
	Image segmenta tion	Supported You can import unlabeled or labeled data. Format specifications of labeled data: Image Segmentation	Supported You can import unlabeled or labeled data. Format specifications of labeled data: Image Segmentation
Aud io	Sound classificati on	Supported You can import unlabeled or labeled data. Follow the format specifications described in Sound Classification .	Supported You can import unlabeled or labeled data. Format specifications of labeled data: Sound Classification
	Speech labeling	Supported You can import unlabeled data.	Supported You can import unlabeled or labeled data. Format specifications of labeled data: Speech Labeling

Dataset Type	Labeling Type	From an OBS Path	From a Manifest File
	Speech paragraph labeling	Supported You can import unlabeled data.	Supported You can import unlabeled or labeled data. Format specifications of labeled data: Speech Paragraph Labeling
Text	Text classification	Supported You can import unlabeled or labeled data. Format specifications of labeled data: Text Classification	Supported You can import unlabeled or labeled data. Format specifications of labeled data: Text Classification
	Named entity recognition	Supported You can import unlabeled data.	Supported You can import unlabeled or labeled data. Format specifications of labeled data: Named Entity Recognition
	Text triplet	Supported You can import unlabeled data.	Supported You can import unlabeled or labeled data. Format specifications of labeled data: Text Triplet
Video	Video labeling	Supported You can import unlabeled data.	Supported You can import unlabeled or labeled data. Format specifications of labeled data: Video Labeling
Other	Free format	Supported You can import unlabeled data.	N/A
Table	Table	Supported Follow the format specifications described in Tables .	N/A

6.3.2.2 Importing Data from an OBS Path to ModelArts

Prerequisites

- A dataset is available.
- Prepare the data to be imported. For details, see [Specifications for Importing Data from an OBS Directory](#).
- The data to be imported is stored in OBS. The manifest file is stored in OBS. For details, see [Creating an OBS Bucket for ModelArts to Store Data](#)
- The OBS bucket and ModelArts are in the same region and you can operate the bucket.

Importing File Data from an OBS Path

The parameters on the GUI for data import vary according to the dataset type. The following uses a dataset of the image classification type as an example.

1. Log in to the [ModelArts management console](#). In the navigation pane, choose **Asset Management** > **Datasets**.
2. Locate the row that contains the desired dataset and click **Import** in the **Operation** column. Alternatively, click the dataset name to go to the **Dashboard** tab page of the dataset, and click **Import** in the upper right corner.
3. In the **Import** dialog box, configure parameters as follows and click **OK**.
 - **Data Source:** **OBS**
 - **Import Mode:** **Path**
 - **Import Path:** OBS path for storing data
 - **Labeling Status:** **Labeled**
 - **Advanced Feature Settings:** disabled by default

Import by Tag enables the system to automatically obtain the labels of the current dataset. Click **Add Label** to add a label. This parameter is optional. If **Import by Tag** is disabled, you can add or delete labels for imported data when labeling data.

Figure 6-8 Importing data from an OBS path

Import

★ Data Source **OBS** Local file

★ Import Mode **Path** manifest

You can save the dataset file to be imported to the OBS path that you have permission to access. [Labeling file format](#)

★ Import Path

You can import up to 999,999 samples and 1,000 labels.

★ Labeling Status **Unlabeled** **Labeled**

★ Labeling Format

The image and labeled file must be stored in the same directory, with the content in the labeled file used as label names.

File storage structure

- dataset-import-example
 - image1.jpg
 - image1.txt
 - image2.jpg
 - image2.txt

Advanced Feature Settings Import by Tag ☐

OK **Cancel**

After the data is imported, it will be automatically synchronized to the dataset. On the **Datasets** page, click the dataset name to view its details and create a labeling job to label the data.

Labeling Status of File Data

The labeling status can be **Unlabeled** or **Labeled**.

- **Unlabeled:** Only the labeling object (such as unlabeled images or texts) is imported.
- **Labeled:** Both the labeling object and content are imported. Labeling content importing is not supported for datasets in free format.

To ensure that the labeling content can be correctly read, you must store data in strict accordance with the specifications.

If **Import Mode** is set to **Path**, store the data to be imported according to the labeling file specifications. For details, see [Specifications for Importing Data from an OBS Directory](#).

If **Import Mode** is set to **manifest**, the manifest file specifications must be met.

 NOTE

- If the labeling status is set to **Labeled**, ensure that the folder or manifest file complies with the format specifications. Otherwise, the import may fail.
- After the import of labeled data, check whether the imported data is in the labeled state.

Importing a Table Dataset from OBS

ModelArts allows you to import table data (CSV files) from OBS.

Import description:

- The prerequisite for successful import is that the schema of the data source must be the same as that specified during dataset creation. The schema indicates column names and types of a table. Once specified during dataset creation, the values cannot be changed.
- When a CSV file is imported from OBS, the data type is not validated, but the number of columns must be the same as that in the schema of the dataset. If the data format is invalid, the data is set to null. For details, see [Table 6-4](#).
- You must select the directory where the CSV file is stored. The number of columns in the CSV file must be the same as that in the dataset schema. The schema of the CSV file can be automatically obtained.

```
dataset-import-example
table_import_1.csv
table_import_2.csv
table_import_3.csv
table_import_4.csv
```

6.3.2.3 Specifications for Importing Data from an OBS Directory

When importing data from OBS, the data storage directory and file name must comply with the ModelArts specifications.

Only the following labeling types of data can be imported by **Labeling Format**: image classification, object detection, image segmentation, text classification, and sound classification.

A table dataset can import data from sources such as OBS and MRS.

 NOTE

- To import data from an OBS directory, you must have the read permission on the OBS directory.
- The OBS buckets and ModelArts must be in the same region.

Image Classification

Data for image classification can be stored in two formats:

Format 1: ModelArts imageNet 1.0

- Images with the same label must be stored in the same directory, with the label name as the directory name. If there are multiple levels of directories, the last level is used as the label name.

In the following example, **Cat** and **Dog** are label names.

```
dataset-import-example
├── Cat
│   ├── 10.jpg
│   ├── 11.jpg
│   └── 12.jpg
└── Dog
    ├── 1.jpg
    ├── 2.jpg
    └── 3.jpg
```

Format 2: ModelArts image classification 1.0

- The image and labeled file must be stored in the same directory, with the content in the labeled file used as label names.

In the following example, **import-dir-1** and **import-dir-2** are the imported subdirectories:

```
dataset-import-example
├── import-dir-1
│   ├── 10.jpg
│   ├── 10.txt
│   ├── 11.jpg
│   ├── 11.txt
│   ├── 12.jpg
│   └── 12.txt
└── import-dir-2
    ├── 1.jpg
    ├── 1.txt
    ├── 2.jpg
    └── 2.txt
```

The following shows a label file for a single label, for example, the **1.txt** file:

```
Cat
```

The following shows a label file for multiple labels, for example, the **2.txt** file:

```
Cat
Dog
```

Only images in JPG, JPEG, PNG, and BMP formats are supported. The size of a single image cannot exceed 5 MB, and the total size of all images uploaded at a time cannot exceed 8 MB.

Object Detection

Data for object detection can be stored in two formats:

Format 1: ModelArts PASCAL VOC 1.0

- The simple mode of object detection requires you to store labeled objects and your label files (in one-to-one relationship with the labeled objects) in the same directory. For example, if the name of the labeled object file is **IMG_20180919_114745.jpg**, the name of the label file must be **IMG_20180919_114745.xml**.

The label files must be in PASCAL VOC format. For details about the format, see [Table 6-14](#).

Example:

```
dataset-import-example
├── IMG_20180919_114732.jpg
├── IMG_20180919_114732.xml
├── IMG_20180919_114745.jpg
├── IMG_20180919_114745.xml
├── IMG_20180919_114945.jpg
└── IMG_20180919_114945.xml
```

A label file example is as follows:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<annotation>
  <folder>NA</folder>
  <filename>bike_1_1593531469339.png</filename>
  <source>
    <database>Unknown</database>
  </source>
  <size>
    <width>554</width>
    <height>606</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>Dog</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <occluded>0</occluded>
    <bndbox>
      <xmin>279</xmin>
      <ymin>52</ymin>
      <xmax>474</xmax>
      <ymax>278</ymax>
    </bndbox>
  </object>
  <object>
    <name>Cat</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <occluded>0</occluded>
    <bndbox>
      <xmin>279</xmin>
      <ymin>198</ymin>
      <xmax>456</xmax>
      <ymax>421</ymax>
    </bndbox>
  </object>
</annotation>
```

Only images in JPG, JPEG, PNG, and BMP formats are supported. A single image cannot exceed 5 MB, and the total size of all images uploaded at a time cannot exceed 8 MB.

Format 2: YOLO

- A YOLO dataset must comply with the following structure:

```
yolo_dataset/
├── obj.names # Label set file
├── obj.data # Files and relative paths for recording dataset information
├── train.txt # Relative path of images in the training set
├── valid.txt # Relative path of images in the validation set
├── obj_train_data/ # Directory where the images in the training set and the corresponding label
files are stored
│   ├── image1.txt # BBox label list for image 1
│   ├── image1.jpg
│   ├── image2.txt
│   ├── image2.jpg
│   └── ...
├── obj_valid_data/ # Directory where the images in the validation set and the corresponding
label files are stored
│   ├── image101.txt
│   └── image101.jpg
```

```

|
|— image102.txt
|— image102.jpg
|— ...

```

A YOLO dataset supports only training sets and validation sets. If other sets are imported, they will be invalid in the YOLO dataset.

- **obj.data** contains the following content and at least one of the **train** and **valid** subsets must be contained. The file paths are relative paths.

```

classes = 5 # Optional
names = <path/to/obj.names># For example, obj.names
train = <path/to/train.txt># For example, train.txt
valid = <path/to/valid.txt># Optional, for example, valid.txt
backup = backup/ # Optional

```

- The **obj.names** file records the label list. Each row label is used as the file index.

```

label1 # index of label 1: 0
label2 # index of label 2: 1
label3
...

```

- The file paths in **train.txt** and **valid.txt** are relative paths, and the file list must be in one-to-one relationship with the files in the directories. The file structures of the two files are as follows:

```

<path/to/image1.jpg># For example, obj_train_data/image.jpg
<path/to/image2.jpg># For example, obj_train_data/image.jpg
...

```

- The .txt files in the **obj_train_data/** and **obj_valid_data/** directories contain the BBox label information of the corresponding images. Each line indicates a BBox label.

```

# image1.txt:
# <label_index> <x_center> <y_center> <width> <height>
0 0.250000 0.400000 0.300000 0.400000
3 0.600000 0.400000 0.400000 0.266667

```

x_center, **y_center**, **width**, and **height** indicate the normalized parameters for the target bounding box: the x-coordinate and y-coordinate of the center point, width, and height.

Only images in JPG, JPEG, PNG, and BMP formats are supported. A single image cannot exceed 5 MB, and the total size of all images uploaded at a time cannot exceed 8 MB.

Image Segmentation

ModelArts image segmentation 1.0:

- Labeled objects and their label files (in one-to-one relationship with the labeled objects) must be in the same directory. For example, if the name of the labeled object file is **IMG_20180919_114746.jpg**, the name of the label file must be **IMG_20180919_114746.xml**.

Fields **mask_source** and **mask_color** are added to the label file in PASCAL VOC format. For details about the format, see [Table 6-10](#).

Example:

```

dataset-import-example
  IMG_20180919_114732.jpg
  IMG_20180919_114732.xml
  IMG_20180919_114745.jpg
  IMG_20180919_114745.xml
  IMG_20180919_114945.jpg
  IMG_20180919_114945.xml

```

A label file example is as follows:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<annotation>
  <folder>NA</folder>
  <filename>image_0006.jpg</filename>
  <source>
    <database>Unknown</database>
  </source>
  <size>
    <width>230</width>
    <height>300</height>
    <depth>3</depth>
  </size>
  <segmented>1</segmented>
  <mask_source>obs://xianao/out/dataset-8153-Jmf5yLjRmSacj9KevS/annotation/V001/segmentationClassRaw/image_0006.png</mask_source>
  <object>
    <name>bike</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <mask_color>193,243,53</mask_color>
    <occluded>0</occluded>
    <polygon>
      <x1>71</x1>
      <y1>48</y1>
      <x2>75</x2>
      <y2>73</y2>
      <x3>49</x3>
      <y3>69</y3>
      <x4>68</x4>
      <y4>92</y4>
      <x5>90</x5>
      <y5>101</y5>
      <x6>45</x6>
      <y6>110</y6>
      <x7>71</x7>
      <y7>48</y7>
    </polygon>
  </object>
</annotation>
```

Text Classification

txt and csv files can be imported for text classification, with the text encoding format of UTF-8 or GBK.

Labeled objects and labels for text classification can be stored in two formats:

- ModelArts text classification combine 1.0: The labeled objects and labels for text classification are in the same text file. You can specify a separator to separate the labeled objects and labels, as well as multiple labels.

For example, the following shows an example text file. The **Tab** key is used to separate the labeled objects from the labels.

```
It touches good and responds quickly. I don't know how it performs in the future. positive
Three months ago, I bought a very good phone and replaced my old one with it. It can operate longer
between charges. positive
Why does my phone heat up if I charge it for a while? The volume button stuck after being pressed
down. negative
It's a gift for Father's Day. The delivery is fast and I received it in 24 hours. I like the earphones
because the bass sounds feel good and they would not fall off. positive
```

- ModelArts text classification 1.0: The labeled objects and labels for text classification are text files, and correspond to each other based on the rows. For example, the first row in a label file indicates the label of the first row in the file of the labeled object.

For example, the content of the labeled object **COMMENTS_20180919_114745.txt** is as follows:

It touches good and responds quickly. I don't know how it performs in the future.
Three months ago, I bought a very good phone and replaced my old one with it. It can operate longer between charges.
Why does my phone heat up if I charge it for a while? The volume button stuck after being pressed down.
It's a gift for Father's Day. The delivery is fast and I received it in 24 hours. I like the earphones because the bass sounds feel good and they would not fall off.

The content of the label file **COMMENTS_20180919_114745_result.txt** is as follows:

positive
negative
negative
positive

This data format requires you to store labeled objects and your label files (in one-to-one relationship with the labeled objects) in the same directory. For example, if the name of the labeled object file is **COMMENTS_20180919_114745.txt**, the name of the label file must be **COMMENTS_20180919_114745_result.txt**.

Example of data files:

```
dataset-import-example
  COMMENTS_20180919_114732.txt
  COMMENTS_20180919_114732_result.txt
  COMMENTS_20180919_114745.txt
  COMMENTS_20180919_114745_result.txt
  COMMENTS_20180919_114945.txt
  COMMENTS_20180919_114945_result.txt
```

Sound Classification

ModelArts audio classification dir 1.0: Sound files with the same label must be stored in the same directory, and the label name is the directory name.

Example:

```
dataset-import-example
  Cat
    10.wav
    11.wav
    12.wav
  Dog
    1.wav
    2.wav
    3.wav
```

Tables

CSV files can be imported from OBS. Select the directory where the files are stored. The number of columns in the CSV file must be the same as that in the dataset schema. The schema of the CSV file can be automatically obtained.

```
dataset-import-example
  table_import_1.csv
  table_import_2.csv
  table_import_3.csv
  table_import_4.csv
```


6.3.2.4 Importing a Manifest File to ModelArts

Prerequisites

- You have created a dataset.
- Prepare the data to be imported. For details, see [Specifications for Importing Data from an OBS Directory](#).
- You have stored the data to be imported in OBS. You have stored the manifest file in OBS.
- The OBS bucket and ModelArts are in the same region and you can operate the bucket.

Importing File Data from a Manifest File

The parameters for data import vary according to the dataset type. The following uses an image dataset as an example.

1. Log in to the [ModelArts management console](#). In the navigation pane, choose **Asset Management** > **Datasets**.
2. Locate the row that contains the desired dataset and click **Import** in the **Operation** column. Alternatively, you can click the dataset name to go to the **Dashboard** tab page of the dataset, and click **Import** in the upper right corner.
3. In the **Import** dialog box, set the parameters as follows and click **OK**.
 - **Data Source:** **OBS**
 - **Import Mode:** **manifest**
 - **Manifest File:** OBS path for storing the manifest file
 - **Labeling Status:** **Labeled**
 - **Advanced Feature Settings:** disabled by default

Import by Tag The system automatically obtains the labels of the dataset. You can click **Add Label** to add a label. This parameter is optional. If **Import by Tag** is disabled, you can add or delete labels for imported data when labeling data.

Import Only Hard Examples: If this parameter is selected, only the **hard** attribute data of the manifest file is imported.

Figure 6-9 Importing a manifest file

Import

★ Data Source: **OBS** Local file

★ Import Mode: Path **manifest**

The manifest file needs to define the mapping between labeling objects and content.
[Manifest file specifications and examples](#)

★ Manifest File:

You can import up to 999,999 samples and 1,000 labels.

★ Labeling Status: Unlabeled **Labeled**

Advanced Feature Settings

Import Only Hard Examples ☐

Import by Tag ☐

OK Cancel

After the data is imported, it will be automatically synchronized to the dataset. On the **Datasets** page, click the dataset name to view its details and create a labeling job to label the data.

Labeling Status of File Data

The labeling status can be **Unlabeled** or **Labeled**.

- **Unlabeled:** Only the labeling object (such as unlabeled images or texts) is imported.
- **Labeled:** Both the labeling object and content are imported. Labeling content importing is not supported for datasets in free format.

To ensure that the labeling content can be correctly read, you must store data in strict accordance with the specifications.

If **Import Mode** is set to **Path**, store the data to be imported according to the labeling file specifications.

If **Import Mode** is set to **manifest**, the manifest file specifications must be met. For details, see [Specifications for Importing a Manifest File](#).

NOTE

If the labeling status is set to **Labeled**, ensure that the folder or manifest file complies with the format specifications. Otherwise, the import may fail.

6.3.2.5 Specifications for Importing a Manifest File

The manifest file defines the mapping between labeled objects and content. The manifest file import mode means that the manifest file is used for dataset import. The manifest file can be imported from OBS. When importing a manifest file from OBS, ensure that you have the permissions to access the directory where the manifest file is stored.

 NOTE

There are many requirements on the manifest file compilation. Import new data from OBS. Generally, manifest file import is used for data migration of ModelArts in different regions or using different accounts. If you have labeled data in a region using ModelArts, you can obtain the manifest file of the published dataset from the output path. Then you can import the dataset using the manifest file to ModelArts of other regions or accounts. The imported data carries the labeling information and does not need to be labeled again, improving development efficiency.

The manifest file that contains information about the original file and labeling can be used in labeling, training, and inference scenarios. The manifest file that contains only information about the original file can be used in inference scenarios or used to generate an unlabeled dataset. The manifest file must meet the following requirements:

- The manifest file uses the UTF-8 encoding format.
- The manifest file uses the JSON Lines format (jsonlines.org). A line contains one JSON object.

```
{ "source": "/path/to/image1.jpg", "annotation": ... }  
{ "source": "/path/to/image2.jpg", "annotation": ... }  
{ "source": "/path/to/image3.jpg", "annotation": ... }
```

In the preceding example, the manifest file contains multiple lines of JSON object.

- The manifest file can be generated by you, third-party tools, or ModelArts Data Labeling. The file name can be any valid file name. To facilitate the internal use of the ModelArts system, the file name generated by the ModelArts data labeling function consists of the following strings:
DatasetName-VersionName.manifest. For example, **animal-v201901231130304123.manifest**.

Image Classification

```
{  
  "source": "s3://path/to/image1.jpg",  
  "usage": "TRAIN",  
  "hard": "true",  
  "hard-coefficient": 0.8,  
  "id": "0162005993f8065ef47eefb59d1e4970",  
  "annotation": [  
    {  
      "type": "modelarts/image_classification",  
      "name": "cat",  
      "property": {  
        "color": "white",  
        "kind": "Persian cat"  
      },  
      "hard": "true",  
      "hard-coefficient": 0.8,  
      "annotated-by": "human",  
      "creation-time": "2019-01-23 11:30:30"  
    },  
    {  
      "type": "modelarts/image_classification",  
      "name": "animal",  
      "annotated-by": "modelarts/active-learning",  
      "confidence": 0.8,  
      "creation-time": "2019-01-23 11:30:30"  
    }  
  ],  
  "inference-loc": "/path/to/inference-output"  
}
```

Table 6-7 Parameters

Parameter	Mandatory	Description
source	Yes	URI of an object to be labeled. For details about data source types and examples, see Table 6-8 .
usage	No	By default, the parameter value is left blank. Possible values are as follows: • TRAIN: The object is used for training. • EVAL: The object is used for evaluation. • TEST: The object is used for testing. • INFERENCE: The object is used for inference. If the parameter value is left blank, you decide how to use the object.
id	No	Sample ID exported from the system. You do not need to set this parameter when importing the sample.
annotation	No	If the parameter value is left blank, the object is not labeled. The value of annotation consists of an object list. For details about the parameters, see Table 6-9 .
inference-loc	No	This parameter is available when the file is generated by the inference service, indicating the location of the inference result file.

Table 6-8 Data source types

Type	Example
OBS	"source": "s3://path-to-jpg"
Content	"source": "content://I love machine learning"

Table 6-9 annotation objects

Parameter	Mandatory	Description
type	Yes	Label type. Possible values are as follows: • image_classification: image classification • text_classification: text classification • text_entity: named entity recognition • object_detection: object detection • audio_classification: sound classification • audio_content: speech labeling • audio_segmentation: speech paragraph labeling
name	Yes/No	This parameter is mandatory for the classification type but optional for other types. This example uses the image classification type.
id	Yes/No	Label ID. This parameter is mandatory for triplets but optional for other types. The entity label ID of a triplet is in E+number format, for example, E1 and E2 . The relationship label ID of a triplet is in R+number format, for example, R1 and R2 .
property	No	Labeling property. In this example, the cat has two properties: color and kind.
hard	No	Indicates whether the example is a hard example. True indicates that the labeling example is a hard example, and False indicates that the labeling example is not a hard example.
annotated-by	No	The default value is human , indicating manual labeling. • human
creation-time	No	Time when the labeling job was created. It is the time when labeling information was written, not the time when the manifest file was generated.
confidence	No	Confidence score of machine labeling. The value ranges from 0 to 1.

Image Segmentation

```
{
  "annotation": [{
    "annotation-format": "PASCAL VOC",
    "type": "modelarts/image_segmentation",
    "annotation-loc": "s3://path/to/annotation/image1.xml",
    "creation-time": "2020-12-16 21:36:27",
    "annotated-by": "human"
  }],
  "usage": "train",
```

```
"source": "s3://path/to/image1.jpg",  
"id": "16d196c19bf61994d7deccafa435398c",  
"sample-type": 0  
}
```

- The parameters such as **source**, **usage**, and **annotation** are the same as those described in [Image Classification](#). For details, see [Table 6-7](#).
- **annotation-loc** indicates the path for saving the label file. This parameter is mandatory for image segmentation and object detection but optional for other labeling types.
- **annotation-format** indicates the format of the label file. This parameter is optional. The default value is **PASCAL VOC**. Only **PASCAL VOC** is supported.
- **sample-type** indicates a sample format. Value **0** indicates image, **1** text, **2** audio, **4** table, and **6** video.

Table 6-10 PASCAL VOC format parameters

Parameter	Mandatory	Description
folder	Yes	Directory where the data source is located
filename	Yes	Name of the file to be labeled
size	Yes	Image pixel • width: image width. This parameter is mandatory. • height: image height. This parameter is mandatory. • depth: number of image channels. This parameter is mandatory.
segmented	Yes	Segmented or not
mask_source	No	Segmentation mask path

Parameter	Mandatory	Description
object	Yes	Object detection information. Multiple object{} functions are generated for multiple objects. • name: type of the labeled content. This parameter is mandatory. • pose: shooting angle of the labeled content. This parameter is mandatory. • truncated: whether the labeled content is truncated (0 indicates that the content is not truncated). This parameter is mandatory. • occluded: whether the labeled content is occluded (0 indicates that the content is not occluded). This parameter is mandatory. • difficult: whether the labeled object is difficult to identify (0 indicates that the object is easy to identify). This parameter is mandatory. • confidence: confidence score of the labeled object. The value ranges from 0 to 1. This parameter is optional. • bndbox: bounding box type. This parameter is mandatory. For details about the possible values, see Table 6-11. • mask_color: label color, which is represented by the RGB value. This parameter is mandatory.

Table 6-11 Bounding box types

Parameter	Shape	Labeling information
polygon	Polygon	Coordinates of points <x1>100<x1> <y1>100<y1> <x2>200<x2> <y2>100<y2> <x3>250<x3> <y3>150<y3> <x4>200<x4> <y4>200<y4> <x5>100<x5> <y5>200<y5> <x6>50<x6> <y6>150<y6> <x7>100<x7> <y7>100<y7>

Example:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<annotation>
  <folder>NA</folder>
  <filename>image_0006.jpg</filename>
  <source>
    <database>Unknown</database>
  </source>
  <size>
    <width>230</width>
    <height>300</height>
    <depth>3</depth>
  </size>
  <segmented>1</segmented>
  <mask_source>obs://xianao/out/dataset-8153-Jmf5yllJmSacj9KevS/annotation/V001/
segmentationClassRaw/image_0006.png</mask_source>
  <object>
    <name>bike</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <mask_color>193,243,53</mask_color>
    <occluded>0</occluded>
    <polygon>
      <x1>71</x1>
      <y1>48</y1>
      <x2>75</x2>
      <y2>73</y2>
      <x3>49</x3>
      <y3>69</y3>
      <x4>68</x4>
      <y4>92</y4>
      <x5>90</x5>
      <y5>101</y5>
      <x6>45</x6>
      <y6>110</y6>
```



```
<x7>71</x7>
<y7>48</y7>
</polygon>
</object>
</annotation>
```

Text Classification

```
{
  "source": "content://I like this product ",
  "id": "XGDVGS",
  "annotation": [
    {
      "type": "modelarts/text_classification",
      "name": "positive",
      "annotated-by": "human",
      "creation-time": "2019-01-23 11:30:30"
    }
  ]
}
```

The **content** parameter indicates the text to be labeled (in UTF-8 encoding format, which can be Chinese). The other parameters are the same as those described in [Image Classification](#). For details, see [Table 6-7](#).

Named Entity Recognition

```
{
  "source": "content://Michael Jordan is the most famous basketball player in the world.",
  "usage": "TRAIN",
  "annotation": [
    {
      "type": "modelarts/text_entity",
      "name": "Person",
      "property": {
        "@modelarts:start_index": 0,
        "@modelarts:end_index": 14
      },
      "annotated-by": "human",
      "creation-time": "2019-01-23 11:30:30"
    },
    {
      "type": "modelarts/text_entity",
      "name": "Category",
      "property": {
        "@modelarts:start_index": 34,
        "@modelarts:end_index": 44
      },
      "annotated-by": "human",
      "creation-time": "2019-01-23 11:30:30"
    }
  ]
}
```

The parameters such as **source**, **usage**, and **annotation** are the same as those described in [Image Classification](#). For details, see [Table 6-7](#).

[Table 6-12](#) describes the property parameters. For example, if you want to extract **Michael** from "**source**":"**content**://**Michael Jordan**", the value of **start_index** is **0** and that of **end_index** is **7**.

Table 6-12 property parameters

Parameter	Data type	Description
@modelarts:start_index	Integer	Start position of the text. The value starts from 0, including the characters specified by start_index .
@modelarts:end_index	Integer	End position of the text, excluding the characters specified by end_index .

Text Triplet

```
{
  "source": "content://"Three Body" is a series of long science fiction novels created by Liu Cix.",
  "usage": "TRAIN",
  "annotation": [
    {
      "type": "modelarts/text_entity",
      "name": "Person",
      "id": "E1",
      "property": {
        "@modelarts:start_index": 67,
        "@modelarts:end_index": 74
      },
      "annotated-by": "human",
      "creation-time": "2019-01-23 11:30:30"
    },
    {
      "type": "modelarts/text_entity",
      "name": "Book",
      "id": "E2",
      "property": {
        "@modelarts:start_index": 0,
        "@modelarts:end_index": 12
      },
      "annotated-by": "human",
      "creation-time": "2019-01-23 11:30:30"
    },
    {
      "type": "modelarts/text_triplet",
      "name": "Author",
      "id": "R1",
      "property": {
        "@modelarts:from": "E1",
        "@modelarts:to": "E2"
      },
      "annotated-by": "human",
      "creation-time": "2019-01-23 11:30:30"
    },
    {
      "type": "modelarts/text_triplet",
      "name": "Works",
      "id": "R2",
      "property": {
        "@modelarts:from": "E2",
        "@modelarts:to": "E1"
      },
      "annotated-by": "human",
      "creation-time": "2019-01-23 11:30:30"
    }
  ]
}
```

The parameters such as **source**, **usage**, and **annotation** are the same as those described in [Image Classification](#). For details, see [Table 6-7](#).

[Table 5 property parameters](#) describes the **property** parameters.

@modelarts:start_index and **@modelarts:end_index** are the same as those of named entity recognition. For example, when **source** is set to **content://"Three Body" is a series of long science fiction novels created by Liu Cix.**, **Liu Cix** is an entity person, **Three Body** is an entity book, the person is the author of the book, and the book is works of the person.

Table 6-13 property parameters

Parameter	Data type	Description
@modelarts:start_index	Integer	Start position of the triplet entities. The value starts from 0, including the characters specified by start_index .
@modelarts:end_index	Integer	End position of the triplet entities, excluding the characters specified by end_index .
@modelarts:from	String	Start entity ID of the triplet relationship.
@modelarts:to	String	Entity ID pointed to in the triplet relationship.

Object Detection

```
{
  "source":"s3://path/to/image1.jpg",
  "usage":"TRAIN",
  "hard":"true",
  "hard-coefficient":0.8,
  "annotation": [
    {
      "type":"modelarts/object_detection",
      "annotation-loc": "s3://path/to/annotation1.xml",
      "annotation-format":"PASCAL VOC",
      "annotated-by":"human",
      "creation-time":"2019-01-23 11:30:30"
    }
  ]
}
```

- The parameters such as **source**, **usage**, and **annotation** are the same as those described in [Image Classification](#). For details, see [Table 6-7](#).
- **annotation-loc** indicates the path for saving the label file. This parameter is mandatory for object detection and image segmentation but optional for other labeling types.
- **annotation-format** indicates the format of the label file. This parameter is optional. The default value is **PASCAL VOC**. Only **PASCAL VOC** is supported.

Table 6-14 PASCAL VOC format parameters

Parameter	Mandatory	Description
folder	Yes	Directory where the data source is located
filename	Yes	Name of the file to be labeled
size	Yes	Image pixel • width: image width. This parameter is mandatory. • height: image height. This parameter is mandatory. • depth: number of image channels. This parameter is mandatory.
segmented	Yes	Segmented or not
object	Yes	Object detection information. Multiple object{} functions are generated for multiple objects. • name: type of the labeled content. This parameter is mandatory. • pose: shooting angle of the labeled content. This parameter is mandatory. • truncated: whether the labeled content is truncated (0 indicates that the content is not truncated). This parameter is mandatory. • occluded: whether the labeled content is occluded (0 indicates that the content is not occluded). This parameter is mandatory. • difficult: whether the labeled object is difficult to identify (0 indicates that the object is easy to identify). This parameter is mandatory. • confidence: confidence score of the labeled object. The value ranges from 0 to 1. This parameter is optional. • bndbox: bounding box type. This parameter is mandatory. For details about the possible values, see Table 6-15.

Table 6-15 Bounding box types

Parameter	Shape	Labeling information
point	Point	Coordinates of a point <x>100<x> <y>100<y>

Parameter	Shape	Labeling information
line	Line	Coordinates of points <x1>100<x1> <y1>100<y1> <x2>200<x2> <y2>200<y2>
bndbox	Rectangle	Coordinates of the upper left and lower right points <xmin>100<xmin> <ymin>100<ymin> <xmax>200<xmax> <ymin>200<ymin>
polygon	Polygon	Coordinates of points <x1>100<x1> <y1>100<y1> <x2>200<x2> <y2>100<y2> <x3>250<x3> <y3>150<y3> <x4>200<x4> <y4>200<y4> <x5>100<x5> <y5>200<y5> <x6>50<x6> <y6>150<y6>
circle	Circle	Center coordinates and radius <cx>100<cx> <cy>100<cy> <r>50<r>

Example:

```
<annotation>
  <folder>test_data</folder>
  <filename>260730932.jpg</filename>
  <size>
    <width>767</width>
    <height>959</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>point</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
```

```
<occluded>0</occluded>
<difficult>0</difficult>
<point>
  <x1>456</x1>
  <y1>596</y1>
</point>
</object>
<object>
  <name>line</name>
  <pose>Unspecified</pose>
  <truncated>0</truncated>
  <occluded>0</occluded>
  <difficult>0</difficult>
  <line>
    <x1>133</x1>
    <y1>651</y1>
    <x2>229</x2>
    <y2>561</y2>
  </line>
</object>
<object>
  <name>bag</name>
  <pose>Unspecified</pose>
  <truncated>0</truncated>
  <occluded>0</occluded>
  <difficult>0</difficult>
  <bndbox>
    <xmin>108</xmin>
    <ymin>101</ymin>
    <xmax>251</xmax>
    <ymax>238</ymax>
  </bndbox>
</object>
<object>
  <name>boots</name>
  <pose>Unspecified</pose>
  <truncated>0</truncated>
  <occluded>0</occluded>
  <difficult>0</difficult>
  <hard-coefficient>0.8</hard-coefficient>
  <polygon>
    <x1>373</x1>
    <y1>264</y1>
    <x2>500</x2>
    <y2>198</y2>
    <x3>437</x3>
    <y3>76</y3>
    <x4>310</x4>
    <y4>142</y4>
  </polygon>
</object>
<object>
  <name>circle</name>
  <pose>Unspecified</pose>
  <truncated>0</truncated>
  <occluded>0</occluded>
  <difficult>0</difficult>
  <circle>
    <cx>405</cx>
    <cy>170</cy>
    <r>100</r>
  </circle>
</object>
</annotation>
```

Sound Classification

```
{
  "source":
```

```
"s3://path/to/pets.wav",
  "annotation": [
    {
      "type": "modelarts/audio_classification",
      "name": "cat",
      "annotated-by": "human",
      "creation-time": "2019-01-23 11:30:30"
    }
  ]
}
```

The parameters such as **source**, **usage**, and **annotation** are the same as those described in [Image Classification](#). For details, see [Table 6-7](#).

Speech Labeling

```
{
  "source": "s3://path/to/audio1.wav",
  "annotation": [
    {
      "type": "modelarts/audio_content",
      "property": {
        "@modelarts:content": "Today is a good day."
      },
      "annotated-by": "human",
      "creation-time": "2019-01-23 11:30:30"
    }
  ]
}
```

- The parameters such as **source**, **usage**, and **annotation** are the same as those described in [Image Classification](#). For details, see [Table 6-7](#).
- The **@modelarts:content** parameter in **property** indicates speech content. The data type is **String**.

Speech Paragraph Labeling

```
{
  "source": "s3://path/to/audio1.wav",
  "usage": "TRAIN",
  "annotation": [
    {
      "type": "modelarts/audio_segmentation",
      "property": {
        "@modelarts:start_time": "00:01:10.123",
        "@modelarts:end_time": "00:01:15.456",
        "@modelarts:source": "Tom",
        "@modelarts:content": "How are you?"
      },
      "annotated-by": "human",
      "creation-time": "2019-01-23 11:30:30"
    },
    {
      "type": "modelarts/audio_segmentation",
      "property": {
        "@modelarts:start_time": "00:01:22.754",
        "@modelarts:end_time": "00:01:24.145",
        "@modelarts:source": "Jerry",
        "@modelarts:content": "I'm fine, thank you."
      },
      "annotated-by": "human",
      "creation-time": "2019-01-23 11:30:30"
    }
  ]
}
```

- The parameters such as **source**, **usage**, and **annotation** are the same as those described in [Image Classification](#). For details, see [Table 6-7](#).
- [Table 6-16](#) describes the **property** parameters.

Table 6-16 **property** parameters

Parameter	Data type	Description
@modelarts:start_time	String	Start time of the sound. The format is hh:mm:ss.SSS . hh indicates the hour, mm indicates the minute, ss indicates the second, and SSS indicates the millisecond.
@modelarts:end_time	String	End time of the sound. The format is hh:mm:ss.SSS . hh indicates the hour, mm indicates the minute, ss indicates the second, and SSS indicates the millisecond.
@modelarts:source	String	Sound source
@modelarts:content	String	Sound content

Video Labeling

```
{
  "annotation": [{
    "annotation-format": "PASCAL VOC",
    "type": "modelarts/object_detection",
    "annotation-loc": "s3://path/to/annotation1_t1.473722.xml",
    "creation-time": "2020-10-09 14:08:24",
    "annotated-by": "human"
  }],
  "usage": "train",
  "property": {
    "@modelarts:parent_duration": 8,
    "@modelarts:parent_source": "s3://path/to/annotation1.mp4",
    "@modelarts:time_in_video": 1.473722
  },
  "source": "s3://input/path/to/annotation1_t1.473722.jpg",
  "id": "43d88677c1e9a971eeb692a80534b5d5",
  "sample-type": 0
}
```

- The parameters such as **source**, **usage**, and **annotation** are the same as those described in [Image Classification](#). For details, see [Table 6-7](#).
- **annotation-loc** indicates the path for saving the label file. This parameter is mandatory for object detection but optional for other labeling types.
- **annotation-format** indicates the format of the label file. This parameter is optional. The default value is **PASCAL VOC**. Only **PASCAL VOC** is supported.
- **sample-type** indicates a sample format. Value **0** indicates image, **1** text, **2** audio, **4** table, and **6** video.

Table 6-17 property parameters

Parameter	Data type	Description
@modelarts:parent_duration	Double	Duration of the labeled video, in seconds
@modelarts:time_in_video	Double	Timestamp of the labeled video frame, in seconds
@modelarts:parent_source	String	OBS path of the labeled video

Table 6-18 PASCAL VOC format parameters

Parameter	Mandatory	Description
folder	Yes	Directory where the data source is located
filename	Yes	Name of the file to be labeled
size	Yes	Image pixel • width: image width. This parameter is mandatory. • height: image height. This parameter is mandatory. • depth: number of image channels. This parameter is mandatory.
segmented	Yes	Segmented or not

Parameter	Mandatory	Description
object	Yes	Object detection information. Multiple object{} functions are generated for multiple objects. • name: type of the labeled content. This parameter is mandatory. • pose: shooting angle of the labeled content. This parameter is mandatory. • truncated: whether the labeled content is truncated (0 indicates that the content is not truncated). This parameter is mandatory. • occluded: whether the labeled content is occluded (0 indicates that the content is not occluded). This parameter is mandatory. • difficult: whether the labeled object is difficult to identify (0 indicates that the object is easy to identify). This parameter is mandatory. • confidence: confidence score of the labeled object. The value ranges from 0 to 1. This parameter is optional. • bndbox: bounding box type. This parameter is mandatory. For details about the possible values, see Table 6-19.

Table 6-19 Bounding box types

Parameter	Shape	Labeling information
point	Point	Coordinates of a point <x>100<x> <y>100<y>
line	Line	Coordinates of points <x1>100<x1> <y1>100<y1> <x2>200<x2> <y2>200<y2>
bndbox	Rectangle	Coordinates of the upper left and lower right points <xmin>100<xmin> <ymin>100<ymin> <xmax>200<xmax> <ymin>200<ymin>

Parameter	Shape	Labeling information
polygon	Polygon	Coordinates of points <x1>100<x1> <y1>100<y1> <x2>200<x2> <y2>100<y2> <x3>250<x3> <y3>150<y3> <x4>200<x4> <y4>200<y4> <x5>100<x5> <y5>200<y5> <x6>50<x6> <y6>150<y6>
circle	Circle	Center coordinates and radius <cx>100<cx> <cy>100<cy> <r>50<r>

Example:

```

<annotation>
  <folder>test_data</folder>
  <filename>260730932_t1.473722.jpg.jpg</filename>
  <size>
    <width>767</width>
    <height>959</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>point</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <occluded>0</occluded>
    <difficult>0</difficult>
    <point>
      <x1>456</x1>
      <y1>596</y1>
    </point>
  </object>
  <object>
    <name>line</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <occluded>0</occluded>
    <difficult>0</difficult>
    <line>
      <x1>133</x1>
      <y1>651</y1>
      <x2>229</x2>
      <y2>561</y2>
    </line>
  </object>
</annotation>

```

```
<object>
  <name>bag</name>
  <pose>Unspecified</pose>
  <truncated>0</truncated>
  <occluded>0</occluded>
  <difficult>0</difficult>
  <bndbox>
    <xmin>108</xmin>
    <ymin>101</ymin>
    <xmax>251</xmax>
    <ymax>238</ymax>
  </bndbox>
</object>
<object>
  <name>boots</name>
  <pose>Unspecified</pose>
  <truncated>0</truncated>
  <occluded>0</occluded>
  <difficult>0</difficult>
  <hard-coefficient>0.8</hard-coefficient>
  <polygon>
    <x1>373</x1>
    <y1>264</y1>
    <x2>500</x2>
    <y2>198</y2>
    <x3>437</x3>
    <y3>76</y3>
    <x4>310</x4>
    <y4>142</y4>
  </polygon>
</object>
<object>
  <name>circle</name>
  <pose>Unspecified</pose>
  <truncated>0</truncated>
  <occluded>0</occluded>
  <difficult>0</difficult>
  <circle>
    <cx>405</cx>
    <cy>170</cy>
    <r>100</r>
  </circle>
</object>
</annotation>
```

6.3.3 Importing Data from MRS to ModelArts

To import data in CSV format stored on HDFS from MRS, select an existing MRS cluster and select the file name or directory from the HDFS file list. The number of columns in the imported file must be the same as that of the dataset schema.

- **Cluster Name:** All MRS clusters of the current account are automatically displayed. However, streaming clusters do not support data import. Select the required cluster from the drop-down list.
- **File Path:** Enter the HDFS file path based on the selected cluster.
- **Contain Table Header:** If this setting is enabled, the imported file contains table headers.

6.3.4 Importing Data from Local Files

Prerequisites

- You have created a dataset.

- You have created an OBS bucket. The OBS bucket and ModelArts are in the same region and you can operate the bucket.

Import Operation

Both file and table data can be uploaded from local files. The data uploaded from local files should be stored in an OBS directory. You must have created an OBS bucket.

In a single batch upload, a maximum of 100 files can be uploaded at a time, and the total size of the files cannot exceed 5 GB.

The parameters on the GUI for data import vary according to the dataset type. The following uses a dataset of the image classification type as an example.

1. Log in to the [ModelArts management console](#). In the navigation pane, choose **Asset Management** > **Datasets**.
2. Locate the row that contains the desired dataset and click **Import** in the **Operation** column.

Figure 6-10 Importing data

Name	Version	Labeling Progress	Created	Description	Operation
def-dataset-image-20200519-001 yfl14z2DMbj256RCKEB	V003	75.00% (12/16)	May 19, 2020 15:56:12 GMT+08:00	--	Import Publish Labeling Export Delete More

Alternatively, you can click the dataset name to go to the **Dashboard** tab page of the dataset, and click **Import** in the upper right corner.

3. In the **Import** dialog box, set the parameters as follows and click **OK**.
 - **Data Source:** **Local file**
 - **Storage Path:** Select an OBS path.
 - **Uploading Data:** Click **Upload data**, upload local data, and click **OK**.

Figure 6-11 Importing data from local files

Import

* Data Source: OBS Local file

* Storage Path: Select an OBS path.

Uploading Data: Upload data

* Labeling Status: Unlabeled Labeled

OK Cancel

6.4 Labeling ModelArts Data

6.4.1 Scenarios

Model training requires a large amount of labeled data. Therefore, label data before training a model. You can create a manual labeling job labeled by one person or by a group of persons (team labeling), or enable auto labeling to quickly label images. You can also modify existing labels, or delete them and re-label.

Model training requires a large amount of labeled data. Therefore, before training a model, label data. ModelArts provides you with the following labeling functions:

- **Manual Labeling:** allows you to manually label data.
- **Auto Labeling:** allows you to automatically label remaining data after a small amount of data is manually labeled.
- **Team Labeling:** allows you to perform collaborative labeling for a large amount of data.

Manual Labeling

Create a labeling job based on the dataset type. ModelArts supports the following types of labeling jobs:

- Image
 - Image classification: identifies a class of objects in images.
 - Object detection: identifies the position and class of each object in an image.
 - Image segmentation: segments an image into different areas based on objects in the image.
- Audio
 - Sound classification: classifies and identifies different sounds.
 - Speech labeling: labels speech content.
 - Speech paragraph labeling: segments and labels speech content.
- Text
 - Text classification: assigns labels to text according to its content.
 - Named entity recognition: assigns labels to named entities, such as time and locations, in text.
 - Text triplet: assigns labels to entity segments and entity relationships in the text.
- Video

Video labeling: identifies the position and class of each object in a video. Only the MP4 format is supported.

Auto Labeling

In addition to manual labeling, ModelArts also provides auto labeling to quickly label data, reducing the labeling time by more than 70%. Auto labeling means learning and training are performed based on the labeled images and an existing model is used to quickly label the remaining images.

This function is supported for only datasets of image classification and object detection.

Team Labeling

Generally, a small data labeling job can be completed by an individual. However, team work is required to label a large dataset. ModelArts provides team labeling, allowing a labeling team that consists of multiple members to manage labels of a dataset.

This function is supported for only datasets of image classification, object detection, text classification, named entity recognition, text triplet, and speech paragraph labeling.

Dataset Functions

Dataset functions vary depending on dataset types. For details, see [Table 6-20](#).

Table 6-20 Functions supported by different types of datasets

Dataset Type	Labeling Type	Manual Labeling	Auto Labeling	Team Labeling
Images	Image classification	Supported	Supported	Supported
	Object detection	Supported	Supported	Supported
	Image segmentation	Supported	-	-
Audio	Sound classification	Supported	-	-
	Speech labeling	Supported	-	-
	Speech paragraph labeling	Supported	-	Supported
Text	Text classification	Supported	-	Supported
	Named entity recognition	Supported	-	Supported
	Text triplet	Supported	-	Supported
Video	Video Labeling	Supported	-	-
Free format	-	-	-	-

Dataset Type	Labeling Type	Manual Labeling	Auto Labeling	Team Labeling
Table	-	-	-	-

6.4.2 Manual Labeling

6.4.2.1 Creating a Manual Labeling Job

Model training requires a large amount of labeled data. Therefore, label data before training a model. You can create a manual labeling job labeled by one person or by a group of persons (team labeling), or enable auto labeling to quickly label images. You can also modify existing labels, or delete them and re-label.

Labeling Job Types

Create a labeling job based on the dataset type. ModelArts supports the following types of labeling jobs:

- Image
 - Image classification: identifies a class of objects in images.
 - Object detection: identifies the position and class of each object in an image.
 - Image segmentation: segments an image into different areas based on objects in the image.
- Audio
 - Sound classification: classifies and identifies different sounds.
 - Speech labeling: labels speech content.
 - Speech paragraph labeling: segments and labels speech content.
- Text
 - Text classification: assigns labels to text according to its content.
 - Named entity recognition: assigns labels to named entities, such as time and locations, in text.
 - Text triplet: assigns labels to entity segments and entity relationships in the text.
- Video

Video labeling: identifies the position and class of each object in a video. Only the MP4 format is supported.

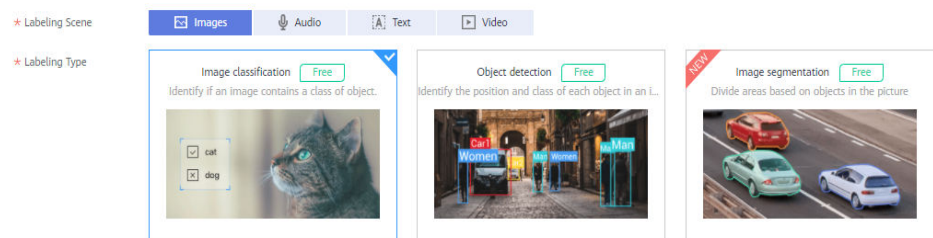
Prerequisites

A dataset of the corresponding type has been created.

Procedure

1. Log in to the ModelArts management console. In the navigation pane on the left, choose **Data Preparation** > **Label Data**.
2. On the **Data Labeling** page, click **Create Labeling Job** in the upper right corner. On the displayed page, create a labeling job.
 - a. Enter basic information about the labeling job, including **Name** and **Description**.
 - b. Select a labeling scene and type as required.

Figure 6-12 Selecting a labeling scene and type



- c. Set the parameters based on the labeling job type. For details, see the parameters of the following labeling job types:
 - **Image Labeling**
 - **Audio Labeling**
 - **Text Labeling**
 - **Video**
- d. Click **Create** in the lower right corner of the page.

After the labeling job is created, the data labeling management page is displayed. You can perform the following operations on the labeling job: start auto labeling, publish new versions, modify the labeling job, and delete the labeling job.

Image Labeling

Figure 6-13 Parameters of labeling jobs for image classification and object detection

The screenshot displays the 'Image Labeling' configuration page in ModelArts. It includes the following elements:

- Name:** A text input field containing 'task-73f8'.
- Description:** A large text area for additional details.
- Labeling Scene:** A tabbed interface with 'Images', 'Audio', 'Text', and 'Video' tabs. 'Images' is currently selected.
- Labeling Type:** Three options are shown: 'Image classification' (with a 'Free' tag and a cat image), 'Object detection' (with a 'Free' tag and a street scene image), and 'Image segmentation' (with a 'Free' tag and a car image).
- Dataset Name:** A dropdown menu showing 'dataset-d1c4' and a 'Create Dataset' button.
- Label Set:** A section for defining labels, including an 'Enter a label name' input, an 'Add Label' button, and a note: 'You can create 9999 more labels.'
- Team Labeling:** A toggle switch currently turned off.

Table 6-21 Parameters of an image labeling job

Parameter	Description
Name	Select a dataset that supports the labeling type.
Label Set	● Label Name: Enter a label name with 1 to 1,024 characters. ● Add Label: Click Add Label to add one or more labels. ● Label Color: Set label colors for object detection and image segmentation labeling jobs. Select a color from the color palette on the right of a label, or enter the hexadecimal color code to set the color. ● Add Label Attribute: For an object detection labeling job, you can click the plus sign (+) on the right to add label attributes after setting a label color. Label attributes are used to distinguish different attributes of the objects with the same label. For example, yellow kittens and black kittens have the same label cat and their label attribute is color.

Parameter	Description
Team Labeling	Enable or disable team labeling. Image segmentation does not support team labeling. Therefore, this parameter is unavailable when you use image segmentation. After enabling team labeling, enter the type of the team labeling job, and select the labeling team and team members. For details about the parameters, see Creating a Team Labeling Job. Before enabling team labeling, ensure that you have added a team and members on the Labeling Teams page. If no labeling team is available, click the link on the page to go to the Labeling Teams page, and add your team and members. For details, see Creating and Managing Teams. After a dataset is created with team labeling enabled, you can view the Team Labeling mark in Labeling Type.

Audio Labeling

Figure 6-14 Parameters of labeling jobs for sound classification, speech labeling, and speech paragraph labeling

* Name

task-73f8

Description

0/256

* Labeling Scene

Images

Audio

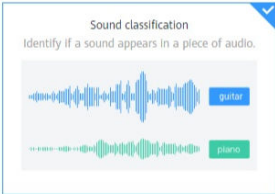
Text

Video

* Labeling Type

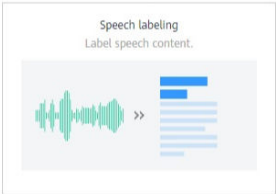
Sound classification

Identify if a sound appears in a piece of audio.



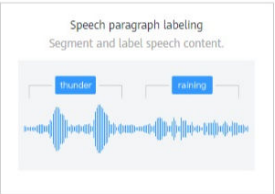
Speech labeling

Label speech content.



Speech paragraph labeling

Segment and label speech content.



* Dataset Name

dataset-d56c

Create Dataset

Label Set

Enter a label name.

Add Label

You can create 9993 more labels.

Table 6-22 Parameters of an audio labeling job

Parameter	Description
Name	Select a dataset that supports the labeling type.
Label Set (for sound classification)	You can add a label set for labeling jobs of sound classification. - Label name: Enter 1 to 1,024 characters in the Label Set text box. Add Label: Click Add Label to add one or more labels.

Parameter	Description
Label Management (for speech paragraph labeling)	Label management is available for speech paragraph labeling. Single Label A single label is used to label a piece of audio that has only one class. Label: Enter a label name with 1 to 1,024 characters. Label Color: Set the label color in the Label Color column. You can select a color from the color palette or enter a hexadecimal color code to set the color. Multiple Labels Multiple labels are suitable for multi-dimensional labeling. For example, you can label a piece of audio as both noise and speech. For speech, you can label the audio with different speakers. You can click Add Label Class to add multiple label classes. A label class can contain multiple labels. The label class or name contains 1 to 256 characters. Only letters, digits, periods (.), underscores (_), and hyphens (-) are allowed. Add Label Class: Enter a label class. Label: Enter a label name. Add Label: Click Add Label to add more labels.
Speech Labeling (for speech paragraph labeling)	Only datasets for speech paragraph labeling support speech labeling. By default, speech labeling is disabled. If this function is enabled, you can label speech content.
Team Labeling (for speech paragraph labeling)	This function is supported for only datasets of speech paragraph labeling. After enabling team labeling, enter the type of the team labeling job, and select the labeling team and team members. For details about the parameters, see Creating a Team Labeling Job. Before enabling team labeling, ensure that you have added a team and members on the Labeling Teams page. If no labeling team is available, click the link on the page to go to the Labeling Teams page, and add your team and members. For details, see Creating and Managing Teams. After a dataset is created with team labeling enabled, you can view the Team Labeling mark in Labeling Type.

Text Labeling

Figure 6-15 Parameters of labeling jobs for text classification, named entity recognition, and text triplet

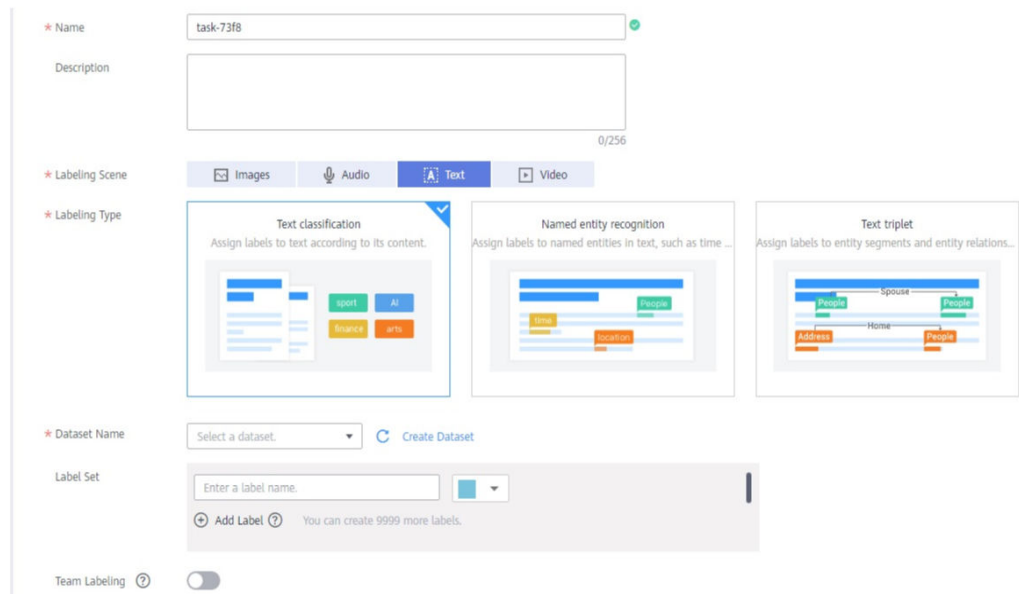
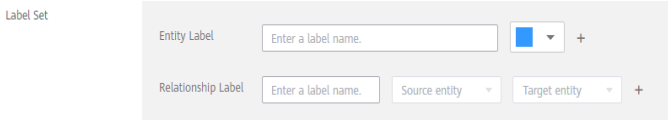


Table 6-23 Parameters of a text labeling job

Parameter	Description
Name	Select a dataset that supports the labeling type.
Label Set (for text classification and named entity recognition)	- Label name: Enter a label name with 1 to 1,024 characters. Add Label: Click Add Label to add one or more labels. - Label color: Select a color from the color palette or enter the hexadecimal color code to set the color. 

Parameter	Description
Label Set (for text triplet)	For datasets of the text triplet type, set entity labels and relationship labels. • Entity Label: Set the label name and label color. You can click the plus sign (+) on the right of the color area to add multiple labels. • Relationship Label: a relationship between two entities. Set the source entity and target entity. Therefore, add at least two entity labels before adding a relationship label. 
Team Labeling	Enable or disable team labeling. After enabling team labeling, enter the type of the team labeling job, and select the labeling team and team members. For details about the parameters, see Creating a Team Labeling Job. Before enabling team labeling, ensure that you have added a team and members on the Labeling Teams page. If no labeling team is available, click the link on the page to go to the Labeling Teams page, and add your team and members. For details, see Creating and Managing Teams. After a dataset is created with team labeling enabled, you can view the Team Labeling mark in Labeling Type.

Video

Figure 6-16 Parameters of a video labeling job

★ Name

task-73f8

Description

0/256

★ Labeling Scene

Images

Audio

Text

Video

★ Labeling Type

Video labeling

Identify the position and class of each object in a vid...
Only MP4 files are supported.



★ Dataset Name

dataset-64ff_v2

Create Dataset

Label Set

Enter a label name.

Add Label

You can create 9997 more labels.

Table 6-24 Parameters of a video labeling job

Parameter	Description
Name	Select a dataset that supports the labeling type.
Label Set	- Label name: Enter a label name with 1 to 1,024 characters. Add Label: Click Add Label to add one or more labels. Label Color: Select a color from the color palette or enter the hexadecimal color code to set the color.

6.4.2.2 Labeling Images

Training a model uses a large number of labeled images. Therefore, label images before the model training. You can add labels to images by manual labeling or auto labeling. In addition, you can modify the labels of images, or remove their labels and label the images again.

Before labeling an image in image classification scenarios, pay attention to the following:

- You can add multiple labels to an image.
- A label name can contain at most 1,024 characters, including letters, digits, hyphens (-), and underscores (_).

Before labeling an image in object detection scenarios, pay attention to the following:

- All target objects in the image must be labeled.
- Target objects are clear without any blocking and contained within bounding boxes.
- Only the entire object can be contained within a bounding box. The bounding box contains the entire object. The edge of the bounding box cannot intersect the edge outline of the object to be labeled. Ensure that there is no gap between the edge and the object to be labeled to prevent the background from interfering with the model training.

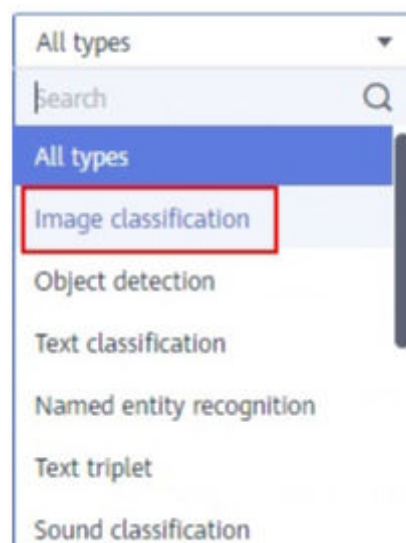
Before labeling an image in image segmentation scenarios, pay attention to the following:

- All objects whose contours need to be extracted from the image must be labeled.
- Polygons can be used for labeling.
 - In polygon labeling, draw a polygon based on the outline of the target object.
- When labeling an image, ensure that the polygons are within the image. Otherwise, an error will occur in subsequent operations.

Starting Labeling

1. Log in to the ModelArts management console. In the navigation pane on the left, choose **Data Preparation > Label Data**.
2. On the right of the labeling job list, select a labeling type from the job type drop-down list. Click the job to be performed based on the labeling type. The details page of the job is displayed.

Figure 6-17 Selecting a labeling type



3. The job details page displays all data of the labeling job.

Image Classification

The labeling job details page displays the **All statuses**, **Unlabeled**, and **Labeled** tabs. The **Unlabeled** tab is displayed by default. Click an image to preview it. For images that have been labeled, the label information is displayed at the bottom of the preview page.

1. In the **Unlabeled** tab, select the images to be labeled.
 - Manual selection: In the image list, click the selection box in the upper left corner of an image to enter the selection mode, indicating that the image is selected. You can select multiple images of the same type and add labels to them together.
 - Batch selection: If all the images on the current page of the image list belong to the same type, you can click **Select Images on Current Page** in the upper right corner to select all the images on the current page.
2. Add labels to the selected images.
 - a. In the label adding area on the right, set a label in the **Label** text box. Click the **Label** text box and select an existing label from the drop-down list. If the existing labels cannot meet the requirements, input a label in the text box.
 - b. Click **OK**. The selected images are automatically moved to the **Labeled** tab page. In the **Unlabeled** and **All statuses** tabs, the labeling information is updated along with the labeling process, including the added label names and the number of images for each label.

Figure 6-18 Adding a label

The figure shows two screenshots from the ModelArts interface. The top screenshot is a dialog box titled 'Add label'. It has a section 'Selected Images' with the text 'Selected 1 image'. Below this is a 'Label' input field containing the placeholder text 'Input a label and press Enter.'. Under the input field is a button labeled 'Add Label' and a small tag that says 'rose' with a close icon. At the bottom of the dialog are 'OK' and 'Cancel' buttons. The bottom screenshot shows a table titled 'All Labels 4' with a 'Label Management' link. The table has two columns: 'Name' and 'Labels'. It lists four labels: 'flower' (0), 'pink' (1), 'rose' (1), and 'sunflower' (2).

Name	Labels
flower	0
pink	1
rose	1
sunflower	2

NOTE

For details about how to label data, see **Labeling Description** on the dataset details page.

1. Log in to the ModelArts management console. In the navigation pane on the left, choose **Data Preparation > Label Data**. The **Data Labeling** page is displayed.
2. In the **My Creations** or **My Participations** tab, locate the dataset to be labeled.
3. Click the dataset name. The labeling details page is displayed. The **Unlabeled** tab is displayed by default.
4. Click **Labeling Description** in the upper right corner.

Object Detection

The labeling job details page displays the **All statuses**, **Unlabeled**, and **Labeled** tabs. The **Unlabeled** tab is displayed by default.

1. In the **Unlabeled** tab, click an image. The system automatically directs you to the page for labeling the image. For details about how to use common buttons on this page, see [Table 6-26](#).
2. In the tool bar, select a proper labeling shape. The default labeling shape is a rectangle. In this example, the rectangle is used for labeling.

 NOTE

In the tool bar, multiple tools are provided for you to label images. After you select a shape to label the first image, the shape automatically applies to subsequent images. You can switch the shape as required.

Table 6-25 Supported bounding boxes

Icon	Description
	Rectangle. You can also press 1 . Click the edge of the upper left corner of the object to be labeled. A rectangle will be displayed. Drag the rectangle to cover the object and click to label the object.
	Polygon. You can also press 2 . In the area where the object to be labeled is located, click to label a point, move the mouse and click multiple points along the edge of the object, and then click the first point again. All the points form a polygon. In this way, the object to be labeled is within the bounding box.
	Round. You can also press 3 . Click the center point of an object, and move the mouse to draw a circle to cover the object and click to label the object.
	Straight. You can also press 4 . Click to specify the start and end points of an object, and move the mouse to draw a straight line to cover the object and click to label the object.
	Dashed line. You can also press 5 . Click to specify the start and end points of an object, and move the mouse to draw a dashed line to cover the object and click to label the object.
	Dot. You can also press 6 . Click the object in an image to label a point.

3. In the **Add Label** text box, enter a new label name, select the label color, and click **Add**. Alternatively, select an existing label from the drop-down list.
Label all objects in an image. Multiple labels can be added to an image. After labeling an image, click the right arrow in the upper right corner of the image (or press **D**) to switch to the next image.
4. Click **Back to Data Labeling Preview** in the upper left corner of the page to view the labeling information. In the displayed dialog box, click **Yes** to save the labeling settings.

The selected images are automatically moved to the **Labeled** tab. In the **Unlabeled** and **All** tabs, the labeling information is updated along with the labeling process, including the added label names and the number of images for each label.

Table 6-26 Common icons on the labeling page

Icon	Description
	Cancel the previous operation. You can also press Ctrl+Z .
	Redo the previous operation. You can also press Ctrl+Shift+Z .
	Zoom in an image. You can also use the scroll wheel.
	Zoom out an image. You can also use the scroll wheel.
	Delete all bounding boxes on the current image. You can also press Shift+Delete .
	Show or hide a bounding box. This operation can be performed only on a labeled image. You can also press Shift+H .
	Drag a bounding box to another position or drag the edge of the bounding box to resize it. You can also use X + left mouse button.
	Reset a bounding box. After dragging a bounding box, you can click this button to quickly restore the bounding box to its original shape and position. You can also press Esc .

Image Segmentation

The labeling job details page displays the **All statuses**, **Unlabeled**, and **Labeled** tabs. The **Unlabeled** tab is displayed by default.

1. In the **Unlabeled** tab, click an image. The system automatically directs you to the page for labeling the image. For details about how to use common buttons on this page, see [Table 6-28](#).
2. Select a labeling method.

On the labeling page, common labeling methods and buttons are provided in the toolbar. By default, polygon labeling is selected. For details, see [Table 6-27](#) and [Table 6-28](#). Use polygon or pole labeling as needed.

NOTE

After you select a method to label the first image, the labeling method automatically applies to subsequent images.

Table 6-27 Labeling methods

Icon	Description
	Polygon. In the area where the object to be labeled is located, click to label a point, move the mouse and click multiple points along the edge of the object, and then click the first point again. All the points form a polygon. In this way, the object to be labeled is within the bounding box.

Table 6-28 Toolbar buttons

Button	Description
	Cancel the previous operation.
	Redo the previous operation.
	Zoom in an image.
	Zoom out an image.
	Delete all bounding boxes on the current image.
	Show or hide a bounding box. This operation can be performed only on a labeled image.
	Drag a bounding box to another position or drag the edge of the bounding box to resize it.
	Reset a bounding box. After dragging a bounding box, you can click this button to quickly restore the bounding box to its original shape and position.
	Display the labeled image in full screen.

3. Label an object.

Identify an object in an image. Click the top, bottom, leftmost, and rightmost points on the object contour. In the displayed dialog box, set the label name and click **Add**. After labeling an image, click  below the image to view the image in the list, and click an unlabeled image for labeling.

4. Click **Back to Data Labeling Preview** in the upper left corner of the page to view the labeling information. In the displayed dialog box, click **Yes** to save the labeling settings.

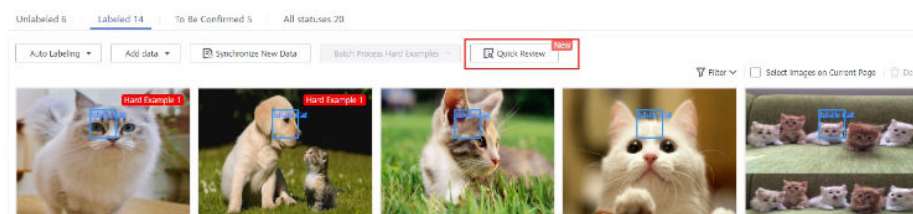
The selected images are automatically moved to the **Labeled** tab. In the **Unlabeled** and **All** tabs, the labeling information is updated along with the labeling process, including the added label names and the number of images for each label.

Quick Review

Current labeling jobs cannot be reviewed in batches. If the label of a sample is modified or deleted, you need to go to the label details page to operate, which is complex. To simplify the operations, you can now review or modify labeling information in batches, improving efficiency.

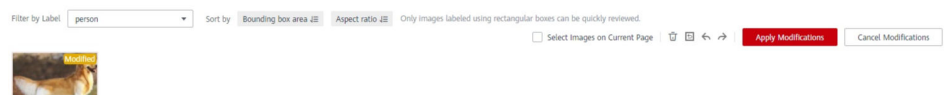
1. Log in to the ModelArts management console. In the navigation pane on the left, choose **Data Preparation > Label Data**. In the **My Creations** tab, select the target labeling job type from the **All types** drop-down list in the upper right corner. (Only object detection and image segmentation support quick review.)
2. In the labeling job list, click the target labeling job. The labeling details page is displayed.
3. Click **Quick Review** in the **Labeled** tab. On the displayed page, confirm the labeling results.

Figure 6-19 Quick review



4. Batch review images of the same label.
 - a. On the review page, select the label type from the drop-down list next to **Filter by Label**.
 - b. Sort images of the selected label type by bounding box area or aspect ratio.
 - c. Click an incorrectly labeled image, and then drag the labeling box to relabel the image. (**Modified** is displayed on the modified images.)
 - d. You can select the incorrectly labeled images, and then click  in the upper right corner to delete the label. (**Deleted** is displayed on the images whose label has been deleted.)

Figure 6-20 Modified



- e. You can also modify the label of a labeled image.
 - i. Select the target images and click  in the **All Labels** area on the right.
 - ii. Type a new label and click **OK**.

Figure 6-21 All labels

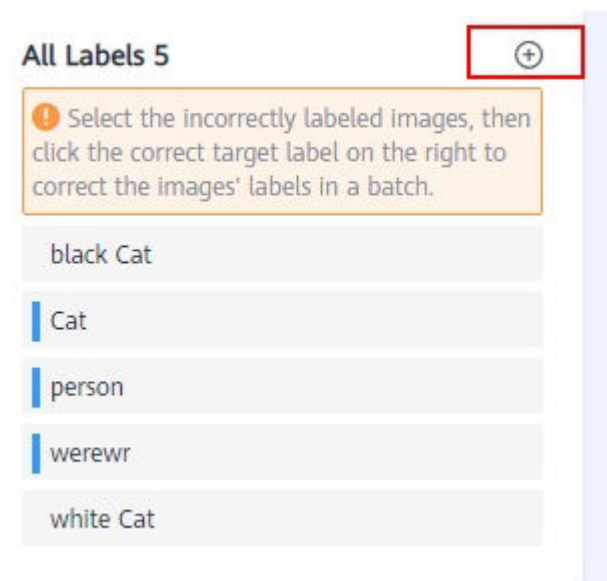
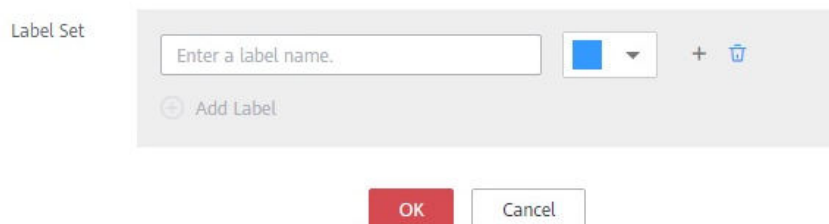


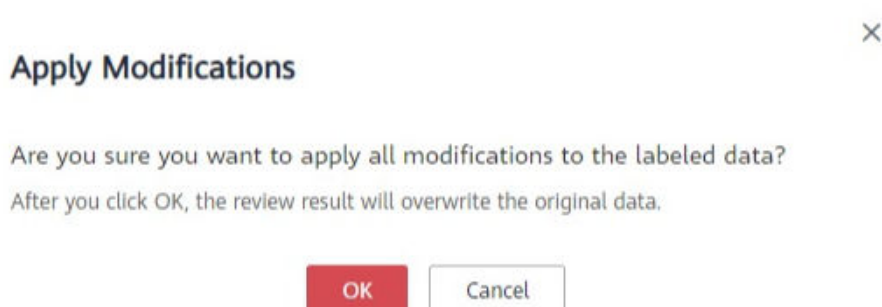
Figure 6-22 Adding a label

Add Label



5. After the modification, click **Apply Modifications**. In the displayed dialog box, click **OK**. The system automatically returns to the labeling overview page and overwrites the original labeling data.

Figure 6-23 Applying modifications



6. If you are not satisfied with the modified data, you can click **Cancel Modifications** to retain the original labeling data.

Figure 6-24 Canceling modifications

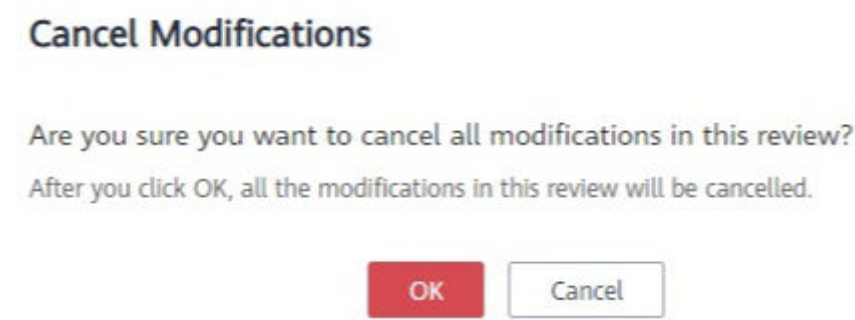


Table 6-29 Buttons on the quick review page

Icon	Description
	Delete the label.
	Undo all operations on the current page.
	Undo the previous operation.
	Redo the previous operation.

6.4.2.3 Labeling Text

Model training requires a large amount of labeled data. Therefore, before the model training, add labels to the files that are not labeled. In addition, you can modify, delete, and re-label the labeled text.

Text classification classifies text content based on labels. Before labeling text content, pay attention to the following:

- Text labeling supports multiple labels. That is, you can add multiple labels to a labeling object.
- A label name can contain a maximum of 1,024 characters, including letters, digits, hyphens (-), underscores (_), and special characters.

Named entity recognition assigns labels to named entities in text, such as time and locations. Before labeling, pay attention to the following:

- A label name of a named entity can contain a maximum of 1,024 characters, including letters, digits, hyphens (-), underscores (_), and special characters.

Triplet labeling is suitable for scenarios where structured information, such as subjects, predicates, and objects, needs to be labeled in statements. With this function, not only entities in statements, but also relationships between entities can be labeled. Triplet labeling is often used in natural language processing tasks such as dependency syntax analysis and information extraction. Before labeling, pay attention to the following:

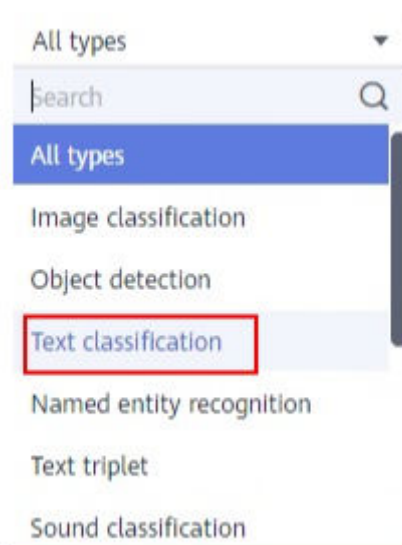
- Ensure that the entity label and relationship label of the labeling job are defined. For **Relationship Label**, set its **Source entity** and **Target entity**. **Relationship Label** must be between the defined **Source entity** and **Target entity**.
- You can define multiple entity and relationship labels for a text object.
- The **Entity Label** defined during dataset creation cannot be deleted.

For example, if two entities are labeled as **Place**, you cannot add any relationship label between them. If a relationship label cannot be added, a red cross is displayed.

Starting Labeling

1. Log in to the ModelArts management console. In the navigation pane on the left, choose **Data Preparation > Label Data**.
2. In the labeling job list, select a labeling type from the **All type** drop-down list, click the job to be performed based on the labeling type. The details page of the job is displayed.

Figure 6-25 Selecting a labeling type



3. The job details page displays all data of the labeling job.

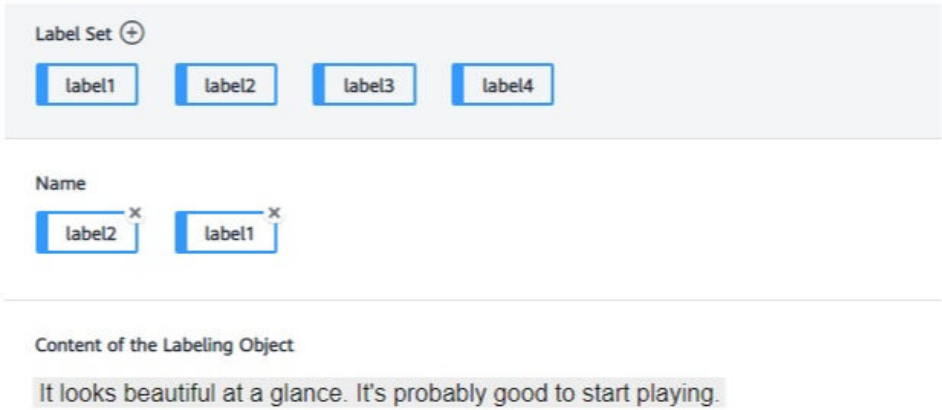
Labeling Text (Text Classification)

The labeling job details page displays the **Unlabeled** and **Labeled** tabs. The **Unlabeled** tab is displayed by default.

1. In the **Unlabeled** tab, the objects to be labeled are listed in the left pane. In the list, click the text object to be labeled, and select a label in the **Label Set** area in the right pane. Multiple labels can be added to a labeling object.

You can repeat this operation to select objects and add labels to the objects.

Figure 6-26 Labeling for text classification



2. After all objects are labeled, click **Save Current Page** at the bottom of the page.

Adding a Tag (Text Classification)

- Adding labels on the **Unlabeled** tab page: Click the plus sign (+) next to **Label Set**. On the **Add Label** page that is displayed, add a label name, select a label color, and click **OK**.

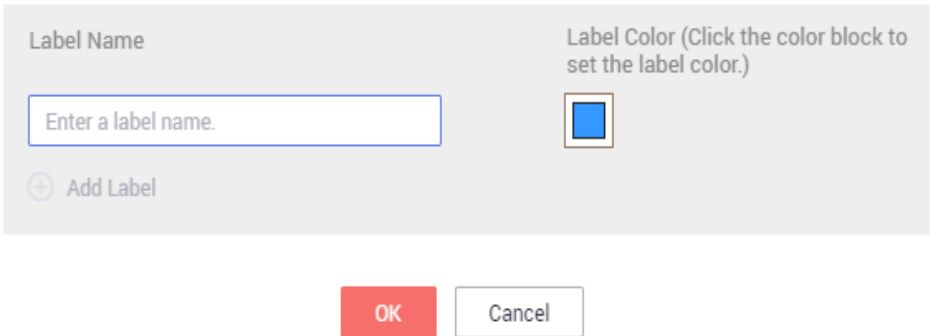
Figure 6-27 Adding a label (1)



- Adding labels in the **Labeled** tab: Click the plus sign (+) next to **Label Set**. On the displayed **Add Label** page, add a label name, select a label color, and click **OK**.

Figure 6-28 Adding a tag

Add Label



Labeling Text (Named Entity Recognition)

The labeling job details page displays the **Unlabeled** and **Labeled** tabs. The **Unlabeled** tab is displayed by default.

1. In the **Unlabeled** tab, the objects to be labeled are listed in the pane on the left. In the list, click the text object to be labeled, select a part of text displayed under **Label Set** for labeling, and select a label in the **Label Set** area in the right pane.

You can repeat this operation to select objects and add labels to the objects.

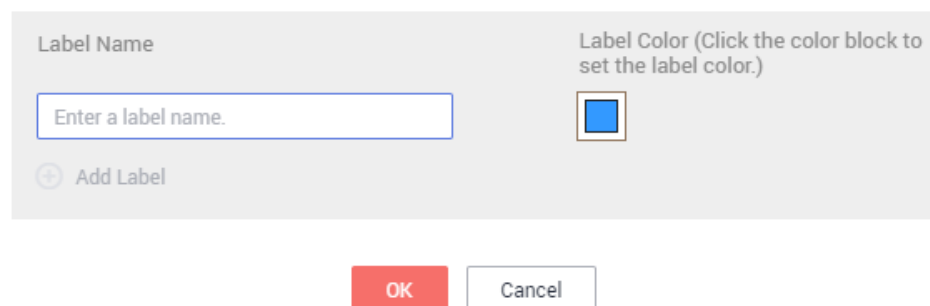
2. Click **Save Current Page** in the lower part of the page to complete the labeling.

Adding a Tag (Named Entity Recognition)

- Adding labels in the **Unlabeled** tab: Click the plus sign (+) next to **Label Set**. On the displayed **Add Label** page, add a label name, select a label color, and click **OK**.
- Adding labels in the **Labeled** tab: Click the plus sign (+) next to **Label Set**. On the displayed **Add Label** page, add a label name, select a label color, and click **OK**.

Figure 6-29 Adding a named entity label

Add Label



The screenshot shows a light gray dialog box titled "Add Label". Inside, there are two main sections. The left section is labeled "Label Name" and contains a text input field with the placeholder "Enter a label name.". Below this input field is a button with a plus sign icon and the text "Add Label". The right section is labeled "Label Color (Click the color block to set the label color.)" and contains a small blue square color block. At the bottom of the dialog box, there are two buttons: a red "OK" button and a white "Cancel" button with a gray border.

Labeling Text (Text Triplet)

The labeling job details page displays the **Unlabeled** and **Labeled** tabs. The **Unlabeled** tab is displayed by default.

1. In the **Unlabeled** tab, the objects to be labeled are listed in the pane on the left. In the list, click a text object, select the corresponding text content on the right pane, and select an entity name from the displayed entity list to label the content.
2. After labeling multiple entities, click the source entity and target entity in sequence and select a relationship type from the displayed relationship list to label the relationship.
3. After all objects are labeled, click **Save Current Page** at the bottom of the page.

 NOTE

You cannot modify the labels of a dataset in the text triplet type on the labeling page. Instead, click **Label Management** and modify the **Entity Label** and **Relationship Label**.

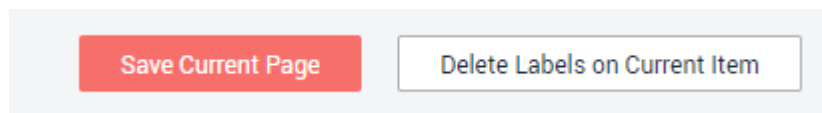
Modify Label (Text Triplet)

After labeling data, you can modify labeled data in the **Labeled** tab.

On the labeling job details page, click the **Labeled** tab. Select a text object in the left pane and the right pane displays the detailed label information. You can move your cursor to the entity or relationship label, and right-click to delete it. You can also click the source entity and target entity in sequence to add a relationship label.

You can click **Delete Labels on Current Item** at the bottom of the page to delete all labels in the selected text object.

Figure 6-30 Deleting current labels



6.4.2.4 Labeling Audio

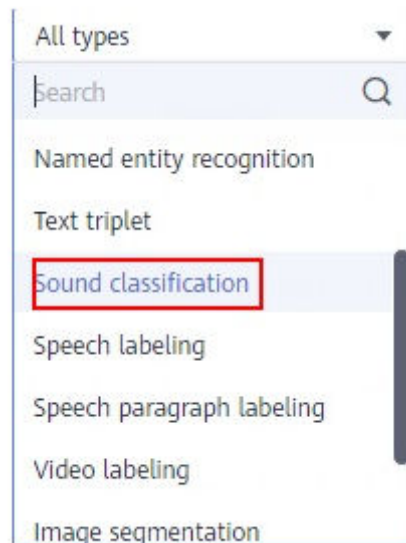
Model training requires a large amount of labeled data. Therefore, before the model training, label the unlabeled audio files. ModelArts enables you to label audio files in batches by one click. In addition, you can modify the labels of audio files, or remove their labels and label the audio files again. Only Chinese and English are supported for audio labeling.

Sound classification is to classify different sounds. Speech labeling is to label speech content. Speech paragraph labeling is to segment and label speech content.

Starting Labeling

1. Log in to the ModelArts management console. In the navigation pane on the left, choose **Data Preparation** > **Label Data**.
2. In the labeling job list, select a labeling type from the **All type** drop-down list, click the job to be performed based on the labeling type. The details page of the job is displayed.

Figure 6-31 Selecting a labeling type



3. The job details page displays all data of the labeling job.

Sound Classification

The labeling job details page displays the **Unlabeled** and **Labeled** tabs. The

Unlabeled tab is displayed by default. Click  on the left of the audio to preview the audio.

1. In the **Unlabeled** tab, select the audio files to be labeled.
 - Manual selection: In the audio list, click the target audio. If the blue check box is displayed in the upper right corner, the audio is selected. You can select multiple audio files of the same type and label them together.
 - Batch selection: If all audio files of the current page belong to one type, you can click **Select Current Page** in the upper right corner of the list to select all the audio files on the page.
2. Add labels.
 - a. In the label adding area on the right, set a label in the **Label** text box.

Method 1 (the required label already exists): In the right pane, select a shortcut from the **Shortcut** drop-down list, select an existing label name from the **Label** text box, and click **OK**.

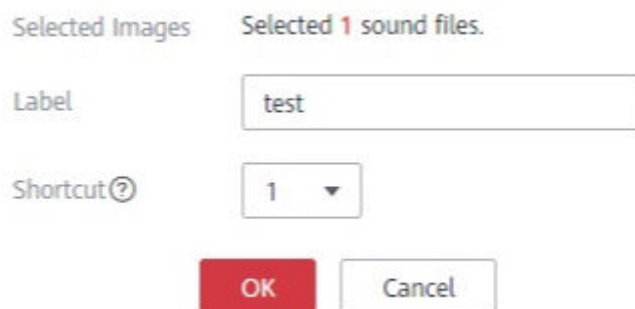
Method 2 (adding a label): In the right pane, select a shortcut from the **Shortcut** drop-down list, and enter a new label name in the **Label** text box.
 - b. The selected audio files are automatically moved to the **Labeled** tab page. On the **Unlabeled** tab page, the labeling information is updated along with the labeling process, including the added label names and the number of audio files corresponding to each label.

NOTE

Shortcut key description: After specifying a shortcut key for a label, you can select an audio file and press the shortcut key to add a label for the audio file. Example: Specify **1** as the shortcut key for the **aa** label. Select one or more files and press **1**. A message is displayed, asking you whether to label the files with **aa**. Click **OK**.

Each label has a shortcut key. A shortcut key cannot be specified for different labels. Shortcut keys can greatly improve the labeling efficiency.

Figure 6-32 Adding an audio label

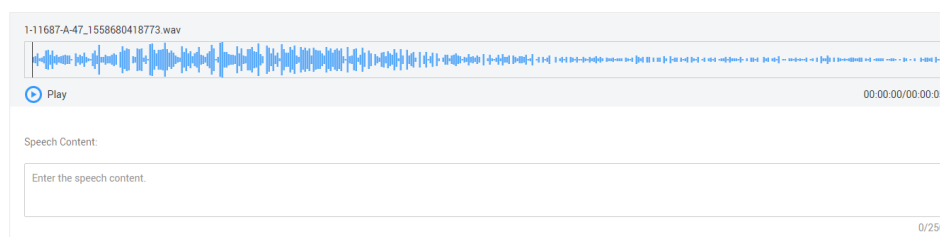


Speech Labeling

The labeling job details page displays the labeled and unlabeled audio files. The **Unlabeled** tab is displayed by default.

1. In the audio file list in the **Unlabeled** tab, click the target audio file. In the area on the right, the audio file is displayed. Click  below the audio file to play the audio.
2. In **Speech Content**, enter the speech content.
3. After entering the content, click **OK** to complete the labeling. The audio file is automatically moved to the **Labeled** tab.

Figure 6-33 Labeling speech content



Speech Paragraph Labeling

The labeling job details page displays the **Unlabeled** and **Labeled** tabs. The **Unlabeled** tab is displayed by default.

1. In the audio file list in the **Unlabeled** tab, click the target audio file. In the area on the right, the audio file is displayed. Click  below the audio file to play the audio.

2. Select an audio segment based on the content being played, and enter the audio file label and content in the **Speech Content** text box.
3. After entering the content, click **Label** to complete the labeling. The audio file is automatically moved to the **Labeled** tab.

6.4.2.5 Labeling Video

Model training requires a large amount of labeled video data. Therefore, before the model training, label the unlabeled video files. ModelArts enables you to label video files. In addition, you can modify the labels of video files, or remove their labels and label the video files again.

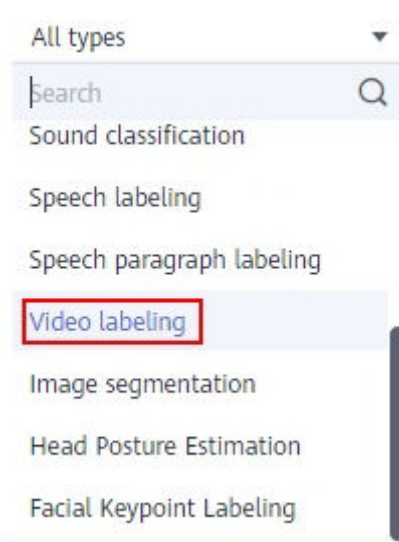
NOTE

Video labeling applies only to video frames.

Starting Labeling

1. Log in to the ModelArts management console. In the navigation pane on the left, choose **Data Preparation > Label Data**.
2. In the labeling job list, select a labeling type from the **All type** drop-down list, click the job to be performed based on the labeling type. The details page of the job is displayed.

Figure 6-34 Selecting a labeling type



3. The job details page displays all data of the labeling job.

Video Labeling

The labeling job details page displays the **Unlabeled**, **Labeled**, and **All statuses** tabs.

1. In the **Unlabeled** tab, click the target video file in the video list on the left. The labeling page is displayed.

2. Play the video. When the video is played to the time point to be labeled, click the pause button in the progress bar to pause the video to a specific image.
3. In the upper pane, select a bounding box. By default, a rectangular box is selected. Drag the mouse to select an object in the video image, enter a new label name in the displayed **Add Label** text box, select a label color, and click **Add** to label the object. Alternatively, select an existing label from the drop-down list and click **Add** to label the object. Label all objects in the image. Multiple labels can be added to an image.

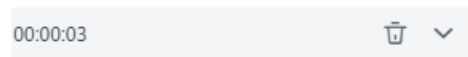
The supported labeling boxes are the same as those for object detection. For details, see [Table 6-26](#).

4. After the previous image is labeled, click the play button on the progress bar to resume the playback. Then, repeat [3](#) to complete labeling on the entire video.

Click **Label List** in the upper right corner of the page. The time points when the video is labeled are displayed.

Figure 6-35 File labels

File Labels



5. Click **Back to Data Labeling Preview** in the upper left corner of the page. The labeling job details page is displayed, and the labeled video file is displayed in the **Labeled** tab.

FAQs

Q: What can I do if the video dataset cannot be displayed or videos cannot be played?

A: If this issue occurs, check the video format. Only MP4 videos can be displayed and played.

6.4.2.6 Managing Labeling Data

Synchronizing New Data

ModelArts automatically synchronizes data and labeling information from datasets to labeling jobs.

To quickly obtain the latest data in a dataset, in the **All statuses**, **Unlabeled**, or **Labeled** tab of the labeling job details page, click **Synchronize New Data**.

NOTE

Symptom:

After the labeled data is uploaded to OBS and synchronized, the data is displayed as unlabeled.

Possible causes:

Automatic encryption is enabled in the OBS bucket.

Solution:

Create an OBS bucket and upload data again, or disable bucket encryption and upload data again.

Filtering Data

In the **All statuses**, **Unlabeled**, or tab, click  in the filter criteria area and add filter criteria to quickly filter the data you want to view.

The following filter criteria are supported. You can set one or more filter criteria.

- **Example Type:** Select **Hard example** or **Non-hard example**.
- **Label:** Select **All** or one or more labels you specified.
- **File Name** or **Path:** Filter files by file name or file storage path.
- **Labeled By:** Select the name of the user who labeled the image.
- **Sample Attribute:** Select the attribute generated by auto grouping. This filter criterion can be used only after auto grouping is enabled.
- **Data Attribute:** Select **All** or **Inference** to filter the data source.

Viewing Labeled Images

On the labeling job details page, click the **Labeled** tab to view the list of labeled images. By default, the corresponding labels are displayed under the image thumbnails. You can also select an image and view the label information of the image in the **Labels of Selected Images** area on the right.

Viewing the Labeled Text

On the dataset details page, click the **Labeled** tab to view the list of the labeled text. You can also view all labels supported by the dataset in the **All Labels** area on the right.

Modifying Labeled Data

After labeling data, you can modify labeled data on the **Labeled** tab page.

- **Modifying based on images**

On the labeling job details page, click the **Labeled** tab, and select one or more images to be modified from the image list. Modify the image information in the label information area on the right.

Modifying a label: In the **Labels of Selected Images** area, click the edit icon in the **Operation** column, enter the correct label name in the text box, and click the check mark to complete the modification.

Deleting a label: In the **Labels of Selected Images** area, click the delete icon in the **Operation** column to delete the label. This operation deletes only the labels added to the selected image.

- **Modifying based on labels**

- On the labeling job details page, click **Label Management**. All labels are displayed on the list.
 - Modifying a label: Click **Modify** in the **Operation** column. In the displayed dialog box, enter a new label name and click **OK**. After the modification, the images with the label added will use the new label name.
 - Deleting a label: Click **Delete** in the **Operation** column to delete the label from all images that have been added with the label.

Figure 6-36 Label Management



The screenshot shows the 'Label Management' interface. At the top, it says 'All Labels 12' and 'Label Management' with a refresh icon. Below is a table with two columns: 'Name' and 'Labels'. The table contains five rows of labels: 'animal' (168), 'cake' (50), 'cat_image' (19), 'data' (293), and 'Data' (6).

Name	Labels
animal	168
cake	50
cat_image	19
data	293
Data	6

- Click **Label** in the **Operation** column of the target labeling job to go to the label management page.
 - Click **Modify** in the **Operation** column of the target label to modify it.
 - Click **Delete** in the **Operation** column of the target label to delete it.

After labeling data, you can modify labeled data in the **Labeled** tab.

- **Modifying based on texts**

On the labeling job details page, click the **Labeled** tab and select the text to be modified from the text list.

In the text list, click the text. When the text background turns blue, the text is selected. If a text file has multiple labels, you can click  on the above to delete the label.

- **Modifying based on labels**

On the labeling job details page, click the **Labeled** tab. The information about all labels is displayed on the right.

- Batch modification: In the **All Labels** area, click the edit icon in the **Operation** column, modify the label name in the text box, select a label color, and click **OK**.

- Batch deletion: In the **All Labels** area, click the deletion icon in the **Operation** column to delete the label. In the displayed dialog box, select **Delete the label** or **Delete the label and objects with only the label**, and click **OK**.

After labeling data, you can modify the labeled data on the **Labeled** tab page.

- **Modifying based on audio**

On the labeling job details page, click the **Labeled** tab. Select one or more audio files to be modified from the audio list. Modify the label in the label details area on the right.

- Modifying a label: In the **File Labels** area, click the edit icon in the **Operation** column, enter the correct label name in the text box, and click the check mark to complete the modification.
- Deleting a label: In the **File Labels** area, click the delete icon in the **Operation** column to delete the label.

- **Modifying based on labels**

On the labeling job details page, click the **Labeled** tab. The information about all labels is displayed on the right.

- Modifying a label: Click the edit icon in the **Operation** column. In the displayed dialog box, enter the new label name and click **OK**. After the modification, the new label applies to the audio files that contain the original label.
- Deleting a label: Click the deletion icon in the **Operation** column. In the displayed dialog box, select the object to be deleted as prompted and click **OK**.

Modifying a Label

After labeling data, you can modify labeled data in the **Labeled** tab.

On the labeling details page, click the **Labeled** tab and then the image to be modified. On the displayed labeling page, modify the labeling information in the **File Labels** area on the right.

- Modifying a label: In the **Labeling** area, click the edit icon, set the target label name or color in the displayed dialog box, and click  to save the modification. Alternatively, click a label to be modified. In the image labeling area, adjust the position and size of the bounding box. After the adjustment is complete, click another label to save the modification.
- Deleting a label: In the **Labeling** area, click the deletion icon to delete a label from the image. After all labels of an image are deleted, the image is displayed in the **Unlabeled** tab.

After the labeling information is modified, click **Back to Data Labeling Preview** in the upper left part of the page to exit the labeling page. In the displayed dialog box, click **Yes** to save the modification.

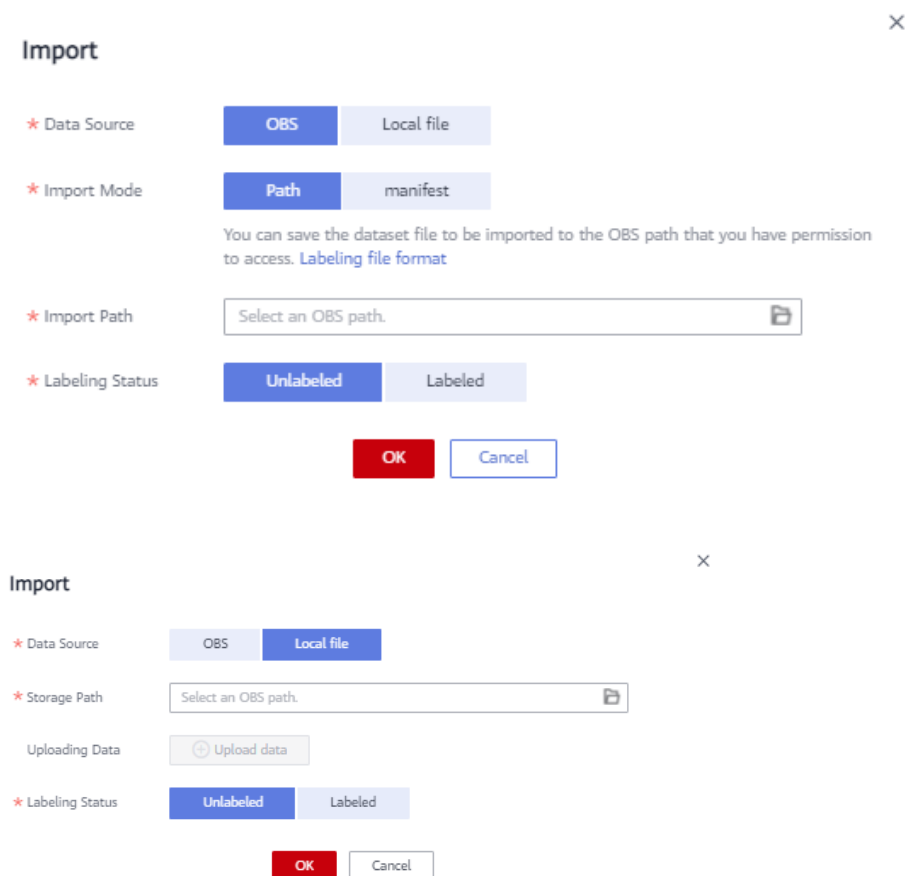
Adding Data

In addition to the data automatically synchronized from datasets, you can directly add images to labeling jobs for labeling. The added data is first imported to the

dataset associated with the labeling task. Then, the labeling task automatically synchronizes the latest data from the dataset.

1. On the labeling job details page, click **All statuses**, **Labeled**, or **Unlabeled** tab, click **Add data** in the upper left corner.
2. Configure input data.

Figure 6-37 Adding images



3. Click **OK**.
- The images you have added will be automatically displayed in the image list in the **All statuses** tab. You can choose **Add data** > **View historical records** to view task history.

Figure 6-38 Viewing historical data

View Task History

Created	Import Mode	Import Path	Samples	Imported Samples	Labeled Samples	Import Stat...
Mar,24,2022 10:21:...	OBS path	obs://el-modelarts-...	19	19	0	✔ Succeed...

Deleting Data

You can quickly delete the files you want to discard.

In the **All statuses**, **Unlabeled**, or **Labeled** tab, select the content to be deleted or click **Select Images on Current Page**, and click **Delete**. In the displayed dialog

box, select or deselect **Delete the source files from OBS** as required. After confirmation, click **OK**.

Figure 6-39 Deleting Images



If a tick is displayed in the upper left corner, the data is selected. If no data is selected on the page, the **Delete** button is unavailable.

NOTE

If you select **Delete the source files from OBS**, images stored in the OBS directory will be deleted accordingly. This operation may affect other dataset versions or datasets using those files, for example, leading to an error in page display, training, or inference. Deleted data cannot be recovered. Exercise caution when performing this operation.

6.4.3 Auto Labeling

6.4.3.1 Creating an Auto Labeling Job

In addition to manual labeling, ModelArts also provides the auto labeling function to quickly label data, reducing the labeling time by more than 70%. Auto labeling means learning and training are performed based on the labeled images and an existing model is used to quickly label the remaining images.

Context

- Only labeling jobs of image classification and object detection types support auto labeling.
- There are at least two types of labels in the labeling job for auto labeling, and each label has been added to at least five images.
- At least one unlabeled image must exist when you enable auto labeling.
- Before starting an auto labeling job, ensure that no auto labeling job is in progress.
- Before starting an auto labeling job, ensure that the image data does not contain any RGBA four-channel images. These images will cause the job to fail. Delete them from the dataset if you find any.

Starting an Auto Labeling Job

1. Log in to the ModelArts management console. In the left navigation pane, choose **Data Preparation > Label Data**. The **Data Labeling** page is displayed.
2. In the labeling job list, locate the target labeling job of the object detection or image classification type, and click **Auto Labeling** in the **Operation** column.
3. On the **Enable Auto Labeling** page, select **Active learning** or **Pre-labeling**. For details, see [Table 6-30](#) and [Table 6-31](#).

Table 6-30 Active learning

Parameter	Description
Auto Labeling Type	Active learning: The system uses semi-supervised learning and hard example filtering to perform auto labeling, reducing manual labeling workload and helping you find hard examples.
Algorithm Type	For a dataset of the image classification type, set the following parameters: Fast: Use the labeled samples for training. Precise: Use labeled and unlabeled samples for semi-supervised training, which improves the model precision.
Specifications	Resource specifications used by an auto labeling job. NOTE Creating an auto labeling job is free, but you will be billed for OBS storage based on usage. For details, see Product Pricing Details . To avoid wasting resources, clear your OBS bucket after labeling jobs and their subsequent tasks are complete.
Compute Nodes	The default value is 1 , indicating the single-node system mode. Only this parameter value is supported.

Table 6-31 Pre-labeling

Parameter	Remarks
Auto Labeling Type	Pre-labeling: Select a model in the My AI Applications tab. Ensure that the model type matches the dataset labeling type. After the pre-labeling is complete, if the labeling result complies with the standard labeling format defined by the platform, the system filters hard examples. This step does not affect the pre-labeling result.
Model and Version	My AI Applications: Select a model as required. Click the drop-down arrow on the left of the target AI application and select a proper version. For details about how to import a model, see Creating a Model
Instance Flavor	In the drop-down list, you can select the node specifications supported by ModelArts.
Compute Nodes	The default value is 1 . You can select a value based on site requirements. The maximum value is 5 .

 NOTE

For labeling jobs of the object detection type, only rectangular boxes can be recognized and labeled when **Active learning** is selected.

Figure 6-40 Enabling auto labeling (image classification)

Enable Auto Labeling

Auto Labeling Type ☒ Active learning ☐ Pre-labeling
The system uses semi-supervised learning and hard example filtering to perform auto labeling, reducing manual labeling workload and helping you find hard examples.

Algorithm Type ☐ ? ☒ Fast ☐ Precise

Specifications GPU: 1*NVIDIA-V100(32GB) | CPU: 8 vCPUs 6... ▼

 To enable Auto Labeling, add at least two classes of labels to the data and use each class of label to label at least 5 images.

Limited-time free
Auto Labeling is billed based on the training duration. [Pricing details](#)

Submit

Figure 6-41 Enabling auto labeling (object detection)

Enable Auto Labeling

Auto Labeling Type ☒ Active learning ☐ Pre-labeling
The system uses semi-supervised learning and hard example filtering to perform auto labeling, reducing manual labeling workload and helping you find hard examples.

Algorithm Type ☐ ? ☒ Fast ☐ Precise

Specifications GPU: 1*NVIDIA-V100(32GB) | CPU: 8 vCPUs 6... ▼

 To enable Auto Labeling, add at least one class of label to the data and ensure that the number of samples with the label is not less than 5. Auto Labeling can identify and add only rectangle labeling boxes.

Limited-time free
Auto Labeling is billed based on the training duration. [Pricing details](#)

Submit

Figure 6-42 Enabling auto labeling (pre-labeling)

×

Enable Auto Labeling

Auto Labeling Type

☐ Active learning

☒ Pre-labeling

Select a model created on the Model Management page. Ensure that the model type matches the dataset labeling type. After the pre-labeling is complete, if the labeling result complies with the standard labeling format defined by the platform, the system filters hard examples. This step does not affect the pre-labeling result.

Model and Version

Select

Specifications

Compute Nodes

—

1

+

Limited-time free

Auto Labeling is billed based on the training duration. [Pricing details](#)

Submit

4. After setting the parameters, click **Submit** to enable auto labeling.

5. In the labeling job list, click a labeling job name to go to the labeling job details page.

6. Click the **To Be Confirmed** tab to view the auto labeling progress.
- You can also enable auto labeling or view the auto labeling history on this tab page.

Figure 6-43 Labeling progress

Labeling | Task Statistics | Label Management

All 40

Unlabeled 0

Labeled 40

To Be Confirmed 0

Filter Criteria

No filter criteria selected.

NOTE

If there are too many auto labeling jobs, they may have to wait in a queue due to limited free resources. This means that they will stay in the labeling state until their turn comes. To ensure that your labeling job can run properly, you are advised to avoid peak hours.

7. After auto labeling is complete, all the labeled images are displayed on the **To Be Confirmed** page.
- Image classification labeling job

On the **To Be Confirmed** page, check whether labels are correct, select the correctly labeled images, and click **OK** to confirm the auto labeling results. The confirmed image will be categorized to the **Labeled** page.

You can modify the labels of the images that are marked as hard examples according to your needs. For details, see [For labeling jobs of the image classification type](#).

- Object detection labeling job

On the **To Be Confirmed** page, click images to view their labeling details and check whether labels and target bounding boxes are correct. For the correctly labeled images, click **Labeled** to confirm the auto labeling results. The confirmed image will be categorized to the **Labeled** page.

You can modify the labels or target bounding boxes of the images marked as hard examples according to your needs. For details, see [For labeling jobs of the object detection type](#).

FAQs

- **What can I do if auto labeling fails?**

Auto labeling is free of charge. If there are too many auto labeling jobs, they may have to wait in a queue due to limited free resources. Create an auto labeling job again or avoid peak hours.

- **What can I do if auto labeling takes a long time?**

Auto labeling is free of charge. If there are too many auto labeling jobs, they may have to wait in a queue due to limited free resources. You are advised to avoid peak hours.

6.4.3.2 Hard Examples of an Auto Labeling Job

In a labeling task that processes a large amount of data, auto labeling results cannot be directly used for training because the labeled images are insufficient at the initial stage of labeling. It takes a lot of time and manpower to adjust and confirm all unlabeled data one by one. To accelerate labeling progress, ModelArts embeds an auto hard example detection function for labeling unlabeled data in an auto labeling task. This function provides suggestions on labeling priorities for remaining unlabeled images. The auto labeling result of an image with high labeling priority is not as expected. Therefore, this case is called a hard example.

The auto hard example detection function is used to automatically label hard examples during auto labeling and data collection and filtering. Further confirm and label hard example data, and add labeling results to the training dataset to obtain a trained model with higher precision. No manual intervention is required for hard example detection, and you only need to confirm and modify the labeled data, improving data management and labeling efficiency. In addition, you can supplement data similar to hard examples to improve the variety of the dataset and further improve the model training precision.

Hard example management involves three scenarios.

- [Confirming Hard Examples After Auto Labeling](#)
- [Labeling Data in a Dataset as Hard Examples](#)

 NOTE

This function is supported for only datasets of image classification and object detection.

Confirming Hard Examples After Auto Labeling

During the execution of an auto labeling task, ModelArts automatically detects and labels hard examples. After the auto labeling task is complete, the labeling results of hard examples are displayed in the **To Be Confirmed** tab. Modify hard example data and confirm the labeling result.

1. Log in to the ModelArts management console. In the navigation pane on the left, choose **Data Preparation > Label Data**. On the **Data Labeling** page, click **My Creations**.
2. In the labeling job list, select a labeling job of the object detection or image classification type and click the labeling job name to go to the labeling job details page.
3. In the **Labeling** tab, click **To Be Confirmed** to check and confirm hard examples.

 NOTE

Labeling data is displayed in the **To Be Confirmed** tab only after the auto labeling task is complete. Otherwise, no data is available in the tab. For details about auto labeling, see [Creating an Auto Labeling Job](#).

- For labeling jobs of the object detection type

In the **To Be Confirmed** tab, click an image to expand its labeling details. Check whether labeling information is correct, for example, whether the label is correct and whether the target bounding box is correctly added to the right position. If the auto labeling result is inaccurate, manually adjust the label or target bounding box and click **Labeled**. Then, the re-labeled data is displayed in the **Labeled** tab.

- For labeling jobs of the image classification type

In the **To Be Confirmed** tab, check whether labels added to images with the **Hard example** mark are correct. Select the images that are incorrectly labeled, delete the incorrect labels, and add correct labels in **Label** on the right. Click **OK**. The selected images and its labeling details are displayed on the **Labeled** tab page.

The selected images are incorrectly labeled. Delete the incorrect labels on the right, add the **dog** label in **Label**, and click **OK** to confirm the hard examples.

Labeling Data in a Dataset as Hard Examples

In a labeling job, labeled or unlabeled image data can be labeled as hard examples. Data labeled as hard examples can be used to improve model precision through built-in rules during subsequent model training.

1. Log in to the ModelArts management console. In the navigation pane, choose **Data Preparation > Label Data**. On the **Data Labeling** page, click **My Creations**.

2. In the labeling job list, select a labeling job of the object detection or image classification type and click the labeling job name to go to the labeling job details page.
3. On the labeling job details page, click the **Labeled**, **Unlabeled**, or **All** tab, select the images to be labeled as hard examples, and choose **Batch Process Hard Examples > Confirm Hard Example**. After the labeling is complete, a **Hard example** mark will be displayed in the upper right corner of a preview image.

6.4.3.3 Auto Grouping for Labeling Jobs

To improve the precision of auto labeling algorithms, you can evenly label multiple classes. ModelArts provides built-in grouping algorithms. You can enable auto grouping to improve data labeling efficiency.

Auto grouping can be understood as data labeling preprocessing. Clustering algorithms are used to cluster unlabeled images, and images are labeled or cleaned by group based on the clustering result.

For example, a user searches for *XX* through a search engine, downloads and uploads related images to the dataset, and then uses the auto grouping function to classify *XX* images, such as papers, posters, images confirmed as *XX*, and others. The user can quickly remove unwanted images from a group or select all images of a type and add labels to the images.

NOTE

This function is supported for only datasets of image classification, object detection, and image segmentation.

Starting Auto Grouping Tasks

1. Log in to the [ModelArts management console](#). In the navigation pane on the left, choose **Data Preparation > Label Data**.
2. In the labeling job list, select a labeling job of the object detection or image classification type and click the labeling job name to go to the labeling job details page.
3. On the **All statuses** tab page of the dataset details page, choose **Auto Grouping > Start Task**.

NOTE

You can start auto group tasks or view task history only on the **All** tab page.

4. In the displayed **Auto Grouping** dialog box, set parameters and click **OK**.
 - **Groups**: Enter an integer from 2 to 200. The parameter value indicates the number of groups into which images are divided.
 - **Result Processing Method**: Select **Update attribute** or **Save to OBS**.
 - **Attribute Name**: If you select **Update attribute**, you need to enter an attribute name.
 - **Result Storage Path**: If you select **Save to OBS**, specify an OBS path.
 - **Advanced Feature Settings**: After this function is enabled, you can select **Clarity**, **Brightness**, and **Color** dimensions for the auto grouping function

so that the grouping is based on the image brightness, color, and clarity.
You can select multiple options.

Figure 6-44 Auto grouping

Auto Grouping

★ Groups

Enter an integer that is less than or equal to the number of samples. If this condition is met, the value ranges from 2 to 200.

★ Result Processing Method

☒ Update attribute

☐ Save to OBS

Add the auto grouping result to the value of an attribute, which can be used as a filter criterion.

★ Attribute Name

★ Advanced Feature Settings ?

☒

☐ Clarity ?

☐ Brightness ?

☐ Color ?

OK

Cancel

5. After the task is submitted, the task progress is displayed in the upper right corner of the page. After the task is complete, you can view the history of the auto grouping tasks to learn task status.

Viewing the Auto Grouping Result

On the **All** tab page of the dataset details page, expand **Filter Criteria**, set **Sample Attribute** to the attribute name of the auto grouping task, and set the sample attribute value to filter the grouping result.

Figure 6-45 Viewing the auto grouping result

Filter Criteria

Label | roses x

+ Add Filter Criterion

Clear All

Example Type

Hard example

Non-hard example

Label

All

dandelion

daisy

roses

sunflowers

tulips

Sample Creation Time

Within 1 month

Within 1 day

Custom

File N...

Enter a keyword and press Enter to create a filter criterion.

X

Labeled By

Select an annotator.

▼

Sample Attribute

-Select-

▼

No attributes available. Click Auto Grouping and select Start Task to create data management attributes.

Viewing Auto Grouping Task History

On the **All** tab page of the dataset details page, choose **Auto Grouping > View Task History**. In the **View Task History** dialog box, basic information about the auto grouping tasks of the current dataset is displayed.

Figure 6-46 Auto grouping task history

View Task History

If the Result Processing Method is Update attribute, you can select attribute values based on the sample attribute as a filter criterion to obtain the result. If the Result Processing Method is Save to OBS, you can view or download the grouping result in the storage path.

Created	Groups	Result Processin...	Storage Path/Att...	Status	Opera...
2020-03-13 09:20...	2	Update attribute	sunflowers	 Running[The j...	Stop

6.4.4 Team Labeling

6.4.4.1 Using Team Labeling

Generally, a small data labeling job can be completed by an individual. However, team work is required to label a large dataset. ModelArts provides team labeling, allowing a labeling team that consists of multiple members to manage labels of a dataset.

NOTE

Team labeling is available only to datasets for image classification, object detection, text classification, named entity recognition, text triplet, and speech paragraph labeling.

For labeling jobs with team labeling enabled, you can create team labeling jobs and assign them to different teams so that team members can complete the labeling jobs together. During data labeling, members can initiate acceptance, continue acceptance, and view acceptance reports.

Team labeling is managed by team. To enable team labeling for a dataset, a team must be specified. Multiple members can be added to a team.

- An account can have a maximum of 10 teams.
- An account must have at least one team to enable team labeling for datasets. If the account has no team, add a team by referring to [Adding a Team](#).

6.4.4.2 Creating and Managing Teams

Team labeling is managed in a unit of teams. To enable team labeling for a dataset, a team must be specified. Multiple members can be added to a team. There is no member in a new team. You need to add members who will participate in a team labeling job.

- An account can have a maximum of 10 teams. A maximum of 100 members can be added to a team. If there are more than 100 members, add them to different teams for better management.

- An account must have at least one team to enable team labeling for datasets. If the account has no team, add a team by referring to [Adding a Team](#).

Adding a Team

1. Log in to the ModelArts management console. In the navigation pane on the left, choose **Data Preparation > Labeling Teams**.
2. On the **Labeling Teams** page, click **Add Team**.
3. In the displayed **Add Team** dialog box, enter a team name and description and click **OK**. The labeling team is added.

The new team is displayed on the **Labeling Teams** page. You can view team details in the right pane. There is no member in the new team. Add members to the new team by referring to [Adding a Member](#).

Adding a Member

1. Log in to the ModelArts management console. In the navigation pane on the left, choose **Data Preparation > Labeling Teams**. The **Labeling Teams** page is displayed.
2. On the **Labeling Teams** page, select a team from the team list on the left and click a team name. The team details are displayed in the right pane.
3. In the **Team Details** area, click **Add Member**.
4. An email address uniquely identifies a team member. Different members cannot use the same email address. The email address you enter will be recorded and saved in ModelArts. It is used only for ModelArts team labeling. After a member is deleted, the email address will also be deleted.

Possible values of **Role** are **Labeler**, **Reviewer**, and **Team Manager**. Only one **Team Manager** can be set.

No annotator cannot be deleted from a labeling team with labeling tasks assigned. The labeling result of an annotator can be synchronized to the overall labeling result only after the annotator's labeling is approved, and the labeling result cannot be filtered.

Information about the added member is displayed in the **Team Details** area.

6.4.4.3 Creating a Team Labeling Job

If you enable team labeling when creating a labeling job and assign a team to label the dataset, the system creates a labeling job based on the team by default. After creating the labeling job, you can view the job in the **My Creations** tab of the dataset.

You can also create a team labeling job and assign it to different members in the same team or to other labeling teams.

Methods

- Choose **Data Preparation > Label Data** on the console. When creating a labeling job, enable **Team Labeling** and select a team or task manager.

Figure 6-47 Enabling team labeling

Team Labeling  

Type ☒ Team ☐ Task Manager

Select Team

team-info 

 Select at least one labeler

Immediately assign the unlabeled files in the dataset to a specified team member for labeling and reviewing. After receiving an email from the system, the team member labels and reviews the files as instructed.

☐ Member Name	Role	Created
No data available.		

☐ Automatically synchronize new files to the team labeling task.
New files in the dataset will be automatically synchronized to the labeling task that has been started.

☐ Automatically load the intelligent labeling results to files that need to be labeled.
Files are automatically labeled. Labelers can then accept or modify the labels.

- Log in to the console and choose **Asset Management > Datasets**. Locate the target dataset and click **Labeling** in the **Operation** column. On the displayed **Create Labeling Job** page, enable **Team Labeling**. You can create multiple team labeling jobs for the same dataset.

Figure 6-48 Enabling team labeling

Team Labeling  

Type ☒ Team ☐ Task Manager

Select Team

team-info 

 Select at least one labeler

Immediately assign the unlabeled files in the dataset to a specified team member for labeling and reviewing. After receiving an email from the system, the team member labels and reviews the files as instructed.

☐ Member Name	Role	Created
No data available.		

☐ Automatically synchronize new files to the team labeling task.
New files in the dataset will be automatically synchronized to the labeling task that has been started.

☐ Automatically load the intelligent labeling results to files that need to be labeled.
Files are automatically labeled. Labelers can then accept or modify the labels.

 NOTE

- Team members receive emails for team labeling jobs. No email will be sent when you create a labeling team or add members to a labeling team. Additionally, after all samples are labeled, no email will be sent when you create a team labeling job.
- After a labeling job is created, all unlabeled samples are assigned to the annotators randomly and non-repeatedly.

Procedure

You can create multiple team labeling jobs for the same dataset and assign them to different members in the same team or to other labeling teams.

1. Log in to the ModelArts management console. In the navigation pane on the left, choose **Asset Management > Datasets**.
2. In the dataset list, select a dataset that supports team labeling, and click the dataset name to go to the **Dashboard** tab of the dataset.
3. In the **Labeling Job** area, view existing labeling jobs of the dataset. Click **Create** to create a job.

Alternatively, you can choose **Asset Management > Label Data** and click **Create Labeling Job**.

4. On the displayed **Create Labeling Job** page, set parameters and click **Create**.
 - **Name**: Enter a job name.
 - **Labeling Scene**: Select the type of the labeling job.
 - **Label Set**: All existing labels and label attributes of the dataset are displayed.
 - **Team Labeling**: Click the button on the right and set the following parameters:
 - **Type**: Select a job type, **Team** or **Task Manager**.
 - **Select Team**: If **Type** is set to **Team**, select a team and members for labeling. The drop-down list displays the labeling teams and their members created by the current account.
 - **Select Task Manager**: If **Type** is set to **Task Manager**, select one **Team Manager** member from all teams as the task manager.
 - **Automatically synchronize new files to the team labeling task**: New files in the dataset will be automatically synchronized to the labeling job that has been started.
 - **Automatically load the intelligent labeling results to files that need to be labeled**: Files are automatically labeled. Annotators can then accept or modify the labels.

 NOTE

The process of loading auto labeling results to a team labeling job is as follows:

- If you set **Type** to **Team**, you are required to create a team labeling task before executing the job.
- If you set **Type** to **Task Manager**, select a team labeling job on the **My Participations** tab page and click **Assign Task**.

Figure 6-49 Creating a team labeling task

Create an label job

Task-offer

Description

0/256

Labeling Source

Images

Audio

Text

Video

Labeling type

Image classification

Free

Identify if an image contains a class of object.

cat

dog



Object detection

Free

Identify the position and class of each object in an i...



Dataset Name

T

Label Set

Enter a label name.

Enter a label name.

Add Label

Team Labeling

Type

Team

Task Manager

Select team

team-info

Select at least one labeler

Immediately assign the unlabeled files in the dataset to a specified team member for labeling and reviewing. After receiving an email from the system, the team member labels and reviews the files as instructed.

Member Name

Role

Created

No data available.

Automatically synchronize new files to the team labeling task.

New files in the dataset will be automatically synchronized to the labeling task that has been started.

Automatically load the intelligent labeling results to files that need to be labeled.

Files are automatically labeled. Labelers can then accept or modify the labels.

Team Labeling

Type

Team

Task Manager

Select Team

team-info

Select at least one labeler

Immediately assign the unlabeled files in the dataset to a specified team member for labeling and reviewing. After receiving an email from the system, the team member labels and reviews the files as instructed.

Member Name

Role

Created

No data available.

Automatically synchronize new files to the team labeling task.

New files in the dataset will be automatically synchronized to the labeling task that has been started.

Automatically load the intelligent labeling results to files that need to be labeled.

Files are automatically labeled. Labelers can then accept or modify the labels.

After the job is created, you can view the new job in the **My Creations** tab.

Issue 01 (2025-08-29)

Copyright © Huawei Cloud Computing Technologies Co., Ltd.

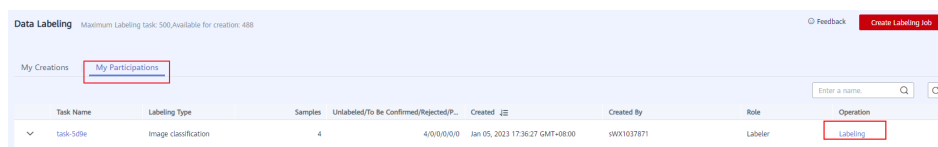
508

Logging In to ModelArts

Typically, users label data in **Data Preparation** of the ModelArts console. ModelArts provides data management capabilities such as dataset management, data labeling, data import and export, auto labeling, and team labeling and management. After a team labeling job is created, team members can log in to the ModelArts console to view the job.

1. After a labeling job is created, receive a labeling notification email as a team member to which the job is assigned.
2. Click the labeling job link in the email. The **Data Preparation > Data Labeling > My Participations** tab on the ModelArts console is displayed. If you have not logged in to the console, log in to it first.
3. In the **My Participations** tab, you can view your labeling jobs.

Figure 6-50 My Participations



The following table lists the data labeling access addresses. If a team member has bound an email address, the team member can receive a job notification email and access the data labeling console using the address provided in the email.

Table 6-32 ModelArts labeling address

Region	URL of the Data Labeling Console
CN-Hong Kong	https://console-intl.huaweicloud.com/modelarts/?region=ap-southeast-1&locale=zh-cn#/dataLabel?tabActive=labelConsole
AP-Singapore	https://console-intl.huaweicloud.com/modelarts/?region=ap-southeast-3&locale=zh-cn#/dataLabel?tabActive=labelConsole
AP-Bangkok	https://console-intl.huaweicloud.com/modelarts/?region=ap-southeast-2&locale=zh-cn#/dataLabel?tabActive=labelConsole

Upon your login, only the team labeling jobs and related data of the current user (the mailbox user) are displayed.

Starting a Team Labeling Job

After logging in to the data labeling page on the management console, you can click the **My Participations** tab to view the assigned labeling job and click the job name to go to the labeling page. The labeling method varies depending on the labeling job type. For details, see the following:

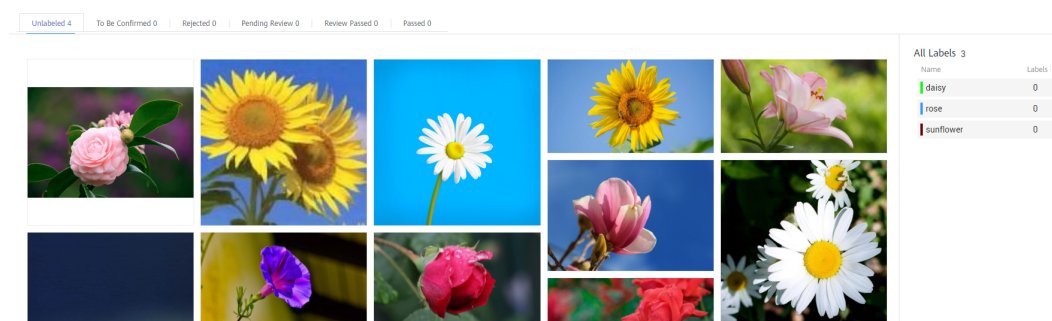
- [Image Classification](#)
- [Object Detection](#)

- [Labeling Text \(Text Classification\)](#)
- [Labeling Text \(Named Entity Recognition\)](#)
- [Labeling Text \(Text Triplet\)](#)
- [Speech Paragraph Labeling](#)

On the labeling page, each member can view the images that are not labeled, to be confirmed, rejected, to be reviewed, approved, and accepted. Pay attention to the images rejected by the administrator and the images to be corrected.

If the Reviewer role is assigned for a team labeling job, the labeling result needs to be reviewed. After the labeling result is reviewed, it is submitted to the administrator for acceptance.

Figure 6-51 Labeling platform



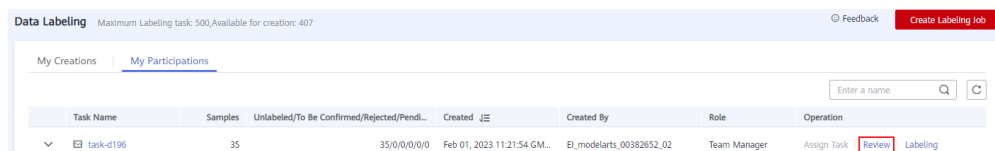
6.4.4.4 Reviewing and Accepting Team Labeling Results

Reviewing Team Labeling Results

After team labeling is complete, the reviewer can review the labeling result.

1. Log in to the ModelArts management console. In the navigation pane on the left, choose **Data Preparation** > **Label Data**. On the **Data Labeling** page, click **My Participations**. Locate the target labeling job and click **Review** in the **Operation** column to initiate the review.

Figure 6-52 Initiating review



2. On the review page, check the samples that are not reviewed, reviewed, approved, or rejected.
3. Choose **Confirm** or **Reject** on the right of the review page.

If you choose **Confirm**, set **Rating** to **A**, **B**, **C**, or **D**. Option **A** indicates the highest score. If you choose **Reject**, enter the rejection reason in the text box.

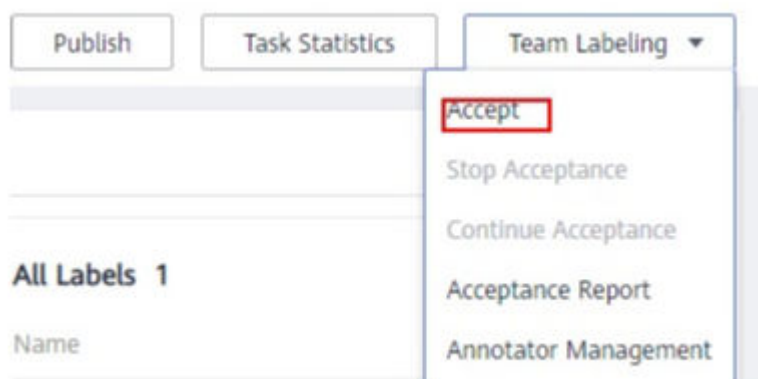
Task Acceptance (Administrator)

- **Initiating acceptance**

After team members complete data labeling, the labeling job creator can initiate acceptance to check labeling results. The acceptance can be initiated only when a labeling member has labeled data. Otherwise, the acceptance initiation button is unavailable.

- a. Log in to the ModelArts management console. In the navigation pane on the left, choose **Data Preparation > Label Data**.
- b. In the **My Participations** tab page, click a team labeling job to go to its details page. Choose **Team Labeling > Accept** in the upper right corner.

Figure 6-53 Initiating acceptance



- c. In the displayed dialog box, set **Sample Policy** to **By percentage** or **By quantity**. Click **OK** to start the acceptance.

By percentage: Sampling is performed based on a percentage for acceptance.

By quantity: Sampling is performed based on quantity for acceptance.

Figure 6-54 Initiating acceptance

Initiate Acceptance

Samples for Acceptance 0

Sampling Policy

☒ By percentage %

☐ By quantity

OK Cancel

- d. After the acceptance is initiated, an acceptance report is displayed on the console. In the **Acceptance Result** area on the right, click **Pass** or **Reject**. If you click **Pass**, set **Rating** to **A**, **B**, **C**, or **D**. Option **A** indicates the highest score. See [Figure 6-55](#). If you click **Reject**, enter your rejection reasons in the text box. See [Figure 6-56](#).

Figure 6-55 Pass

Acceptance Result

Rating: ☒ A ☐ B ☐ C ☐ D

[Skip](#)

Figure 6-56 Reject

Acceptance Result

not good

8/256

[Skip](#)

- Continuing acceptance

You can continue accepting tasks whose acceptance is not completed. For tasks for which an acceptance process is not initiated, the **Continue Acceptance** button is unavailable.

In the **Labeling Progress** pane in the **Task Statistics** tab, click **Continue Acceptance** to continue accepting jobs. The **Real-Time Acceptance Report** page is displayed. You can continue to accept the images that are not accepted.

- Finishing acceptance

After the continue acceptance is complete, click **Stop Acceptance** in the upper right corner. On the displayed page, view the acceptance status of the labeling job, such as the number of sampled files, configure parameters, and perform the acceptance. The labeling information is synchronized to the **Labeled** tab of the labeling job only after the acceptance is complete.

Once the labeled data is accepted, team members cannot modify the labeling information. Only the dataset creator can modify the labeling information.

Table 6-33 Parameters for finishing acceptance

Parameter	Description
Modifying Labeled Data	Not overwrite: For the same data, do not overwrite the existing data with the labeling result of the current team. Overlays: For the same data, overwrite the existing data with the labeling result of the current team. Overwritten data cannot be recovered.

Parameter	Description
Acceptance Scope	• All passed: All items, including the rejected ones will pass the review. • All rejects: All items, including the ones that have passed the review will be rejected. In this case, the passed items must be labeled and reviewed again in the next acceptance. • All remaining items pass: The rejected items are still rejected, and the remaining items will automatically pass the review. • All remaining items rejects: The selected items that have passed the review do not need to be labeled. All the selected items that have been rejected and the items that have not been selected must be labeled again for acceptance.

Viewing an Acceptance Report

You can view the acceptance report of an ongoing or finished labeling job. Log in to the management console and choose **Data Preparation > Label Data**. On the **Data Labeling** page, select **My Creations** and click the name of a team labeling job. The job details page is displayed. In the upper right corner of the page, click **Acceptance Report**. In the displayed dialog box, view report details.

Figure 6-57 Viewing an Acceptance Report

Acceptance Report

Statistics of In Progress Acceptance

Acceptance Pass Rate	— %	File Samples	--	Pending Acceptance	--
Accepted	0	Passed	0	Rejected	-

Statistics of Completed Acceptance

Acceptance Pass Rate	--%	File Samples	--	Pending Acceptance	--
Accepted	--	Passed	--	Rejected	--

OK

Deleting a Labeling Job

After a job is accepted, delete it if the labeling job is no longer used. After a job is deleted, the labeling details that are not accepted will be lost. However, the original data in the dataset and the labeled data that has been accepted are still stored in the corresponding OBS bucket.

6.4.4.5 Managing Teams and Team Members

Modifying Member Information

You can modify member information if it is changed.

1. In the **Team Details** area, locate the target member.
2. Click **Modify** in the **Operation** column. In the displayed dialog box, modify the description or role.

The email address of a member cannot be changed. To change the email address of a member, delete the member, and set a new email address when adding a member.

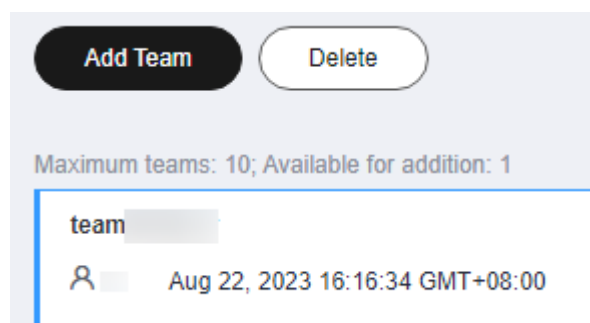
Possible values of **Role** are **Labeler**, **Reviewer**, and **Team Manager**. Only one **Team Manager** can be set.

Deleting a Team

You can delete a team that is no longer used.

On the **Labeling Teams** page, select the target team and click **Delete**. In the displayed dialog box, click **OK**.

Figure 6-58 Deleting a team



Deleting a Member

- **Deleting a single member**

In the **Team Details** area, locate the target member, and click **Delete** in the **Operation** column. In the displayed dialog box, click **OK**.

- **Deleting in batches**

In the **Team Details** area, select members to be deleted and click **Delete**. In the displayed dialog box, click **OK**.

Figure 6-59 Batch deletion

Member Name	ID	Role	Blind	Created	Description	Operation
User Name Account Name	User ID Account ID	Labeler	--	Apr 10, 2024 10:41:42 GM...	--	Modify Delete
User Name Account Name	User ID Account ID	Labeler	--	Aug 23, 2023 15:19:34 GM...	--	Modify Delete

Managing Annotators

If team labeling is enabled for a labeling job, view its labeling details in the **Annotator Management** tab. Additionally, you can add, modify, or delete annotators.

1. Choose **Data Preparation > Label Data**. In the **My Creations** tab, view the list of all labeling jobs.
 2. Locate the target team labeling job. (The name of a team labeling job is followed by )
 3. Choose **More > Annotator Management** in the **Operation** column. Alternatively, click the job name to go to the job details page, and choose **Team Labeling > Annotator Management** in the upper right corner.
- Adding an annotator
Click **Add Member**, select a member name, and click **OK**.
Click **Send Email** in the **Operation** column to send the labeling job to the annotator by email.
 - Modifying annotator information
Click **Modify** in the **Operation** column to modify the role of the annotator.
 - Deleting an annotator
Click **Delete** in the **Operation** column to delete the annotator.

6.4.5 Managing Labeling Jobs

Viewing Labeling Jobs

On the ModelArts **Data Labeling** page, view your created labeling jobs in the **My Creations** tab.

1. Log in to the ModelArts management console. In the navigation pane on the left, choose **Data Preparation > Label Data**.
2. In the **My Creations** tab, you can view the created labeling jobs. You can view information about these labeling jobs.

Figure 6-60 My Creations

Name	Dataset	Labeling Progress (Labeled/Total)	Pending Confir...	Created	Description	Operation
task-1fcd	dataset-c7ce	100% (291/291)	--	Jul 24, 2022 17:32:57 GMT+08:00	--	Auto Labeling Label Publish More
dataset-f606	dataset-f606	0% (0/311)	--	Jul 23, 2022 15:20:49 GMT+08:00	--	Auto Labeling Label Publish More
dataset-d56c	dataset-d56c	4% (1/23)	--	Jul 23, 2022 14:31:31 GMT+08:00	--	Label Publish Task Statistics Modify

3. In the **My Participations** tab, you can view the labeling jobs you have participated in. You can view the labeling job details, including the labeling team members and labeling progress.

Copying a Labeling Job

1. Log in to the ModelArts management console. In the navigation pane on the left, choose **Data Preparation** > **Label Data**. The **Data Labeling** page is displayed.
2. In the **My Creations** tab, locate the labeling job you want to copy.
3. Choose **More** > **Copy** in the **Operation** column of the job.
4. In the **Copy Task** dialog box, enter the job description and job name *Task name-copy-xxxx*, where *xxxx* is a randomly generated code to distinguish the new job from the copied job. You can also change the name of the new job. Click **Yes**.

Copy Task

You are copying the labeling task **dataset-a4d9**. After you click OK, a new labeling task is created based on the task.

Name	dataset-a4d9-copy-c78f
Description	

0/256

 The copy operation rebuilds the job based on the dataset associated with the job. If the dataset sample changes, the new job will change accordingly. In addition, the labeling result cannot be copied.

5. After the labeling job is copied, you can obtain the new labeling job on the labeling job list page. The new labeling job information includes the samples, labels, and team labeling information.

Filtering Data

In the **Dashboard** tab of the dataset, the summary of the dataset is displayed by default. In the upper right corner of the page, click **Label**. The dataset details page is displayed, showing all data in the dataset by default. In the **All**, **Unlabeled**, or **Labeled** tab, you can add filter criteria in the filter criteria area to quickly filter the data you want to view.

The following filter criteria are supported. You can set one or more filter criteria.

- **Example Type:** Select **Hard example** or **Non-hard example**.
- **Label:** Select **All** or one or more labels you specified.
- **Sample Creation Time:** Select **Within 1 month**, **Within 1 day**, or **Custom** to customize a time range.
- **File Name** or **Path:** Filter files by file name or file storage path.
- **Labeled By:** Select the name of the user who labeled the image.
- **Sample Attribute:** Select the attribute generated by auto grouping. This filter criterion can be used only after **auto grouping** is enabled.
- **Data Attribute:** This criterion is not supported.

6.5 Publishing ModelArts Data

ModelArts distinguishes data of the same source according to versions processed or labeled at different time, which facilitates the selection of dataset versions for subsequent model building and development.

About Dataset Versions

- For a newly created dataset (before publishing), there is no dataset version information. The dataset must be published before being used for model development or training.
- The default naming rules of dataset versions are V001 and V002 in ascending order. You can customize the version number during publishing.
- You can set any version to the current directory. Then the details of the version are displayed on the dataset details page.
- You can obtain the dataset in the manifest file format corresponding to each dataset version based on the value of **Storage Path**. The dataset can be used when you import data or filter hard examples.
- The version of a table dataset cannot be changed.

Publishing a Dataset Version

1. Log in to the **ModelArts management console**. In the navigation pane on the left, choose **Asset Management > Datasets**.
2. Locate the target dataset and click **Publish** in the **Operation** column. Alternatively, click the dataset name to go to the **Dashboard** tab of the dataset, and click **Publish** in the upper right corner.
3. In the displayed dialog box, set the parameters and click **OK**.

Table 6-34 Parameters for publishing a dataset

Parameter	Description
Version	The naming rules of V001 and V002 in ascending order are used by default. A version name can be customized. Only letters, digits, hyphens (-), and underscores (_) are allowed.

Parameter	Description
Format	Only table datasets support version format setting. Available values are CSV and CarbonData. NOTE If the exported CSV file contains any command starting with =, +, -, or @, ModelArts automatically adds the Tab setting and escapes the double quotation marks (") for security purposes.
Splitting	Only image classification, object detection, text classification, and sound classification datasets support data splitting. By default, this function is disabled. After this function is enabled, you need to set the training and validation ratios. Enter a value ranging from 0 to 1 for Training Set Ratio. After the training set ratio is set, the validation set ratio is determined. The sum of the training set ratio and the validation set ratio is 1. NOTE To ensure the model accuracy, set the training set ratio to 0.8 or 0.9. The training set ratio is the ratio of sample data used for model training. The validation set ratio is the ratio of the sample data used for model validation. The training and validation ratios affect the performance of training templates.
Description	Description of the current dataset version.
Hard Example	Only image classification and object detection datasets support hard example attributes. By default, this function is disabled. After this function is enabled, information such as the hard example attributes of the dataset are written to the corresponding manifest file.

Directory Structure of Dataset Versions

Datasets are managed based on OBS directories. After a new version is published, the directory is generated based on the new version in the output dataset path.

Take an image classification dataset as an example. After the dataset is published, the directory structure of related files generated in OBS is as follows:

```
-- user-specified-output-path
|-- DatasetName-datasetId
|   |-- annotation
|   |   |-- VersionName1
|   |   |   |-- VersionName1.manifest
|   |   |-- VersionName2
|   |   ...
|   |-- ...
```

The following uses object detection as an example. If a manifest file is imported to the dataset, the following provides the directory structure of related files after the dataset is published:

```
|-- user-specified-output-path
|   |-- DatasetName-datasetId
|       |-- annotation
|           |-- VersionMame1
|               |-- VersionMame1.manifest
|                   |-- annotation
|                       |-- file1.xml
|               |-- VersionMame2
|                   ...
|               |-- ...
```

Take video labeling as an example. After the dataset is published, the labeling result file (XML) is stored in the dataset output directory.

```
|-- user-specified-output-path
|   |-- DatasetName-datasetId
|       |-- annotation
|           |-- VersionMame1
|               |-- VersionMame1.manifest
|                   |-- annotations
|                       |-- images
|                           |-- videoName1
|                               |-- videoName1.timestamp.xml
|                           |-- videoName2
|                               |-- videoName2.timestamp.xml
|           |-- VersionMame2
|               ...
|           |-- ...
```

The key frames for video labeling are stored in the dataset input directory.

```
|-- user-specified-input-path
|   |-- images
|       |-- videoName1
|           |-- videoName1.timestamp.jpg
|       |-- videoName2
|           |-- videoName2.timestamp.jpg
```

Viewing Dataset Version Updates

1. Log in to the ModelArts management console. In the navigation pane on the left, choose **Datasets**.
2. Locate the target dataset and choose **More > Manage Version** in the **Operation** column. The **Manage Version** tab is displayed.

You can view basic information about the dataset, and view the versions and publish time on the left.

Setting to Current Version

1. Log in to the ModelArts management console. In the navigation pane on the left, choose **Datasets**.
2. Locate the target dataset and choose **More > Manage Version** in the **Operation** column. The **Manage Version** tab is displayed.
3. In the **Manage Version** tab, select the target dataset version, and click **Set to Current Version** in the basic information area on the right. After the setting is complete, **Current version** is displayed to the right of the version name.

NOTE

Only the version in **Normal** status can be set to the current version.

Deleting a Dataset Version

1. Log in to the ModelArts management console. In the navigation pane on the left, choose **Datasets**.
2. Locate the target dataset and choose **More > Manage Version** in the **Operation** column. The **Manage Version** tab is displayed.
3. Locate the target version and click **Delete** in the **Operation** column. In the displayed dialog box, click **OK**.

NOTE

Deleting a dataset version does not remove the original data. Data and its labeling information are still stored in the OBS directory. However, this affects version management. Exercise caution when performing this operation.

6.6 Analyzing ModelArts Data Characteristics

Images or target bounding boxes are analyzed based on image features, such as blurs and brightness to draw visualized curves to help process datasets.

You can also select multiple versions of a dataset to view their curves for comparison and analysis.

Context

- Data feature analysis is only available for image datasets of the image classification and object detection types.
- Data feature analysis is only available for the published datasets. The published dataset versions in **Default** format support data feature analysis.
- A data scope for feature analysis varies depending on the dataset type.
 - In a dataset of the object detection type, if the number of labeled samples is 0, the **View Data Feature** tab is unavailable and data features are not displayed after a version is published. After the images are labeled and the version is published, the data features of the labeled images are displayed.
 - In a dataset of the image classification type, if the number of labeled samples is 0, the **View Data Feature** tab is unavailable and data features are not displayed after a version is published. After the images are labeled and the version is published, the data features of all images are displayed.
- The analysis result is valid only when the number of images in a dataset reaches a certain level. Generally, more than 1,000 images are required.
- Image classification supports the following data feature metrics: **Resolution**, **Aspect Ratio**, **Brightness**, **Saturation**, **Blur Score**, and **Colorfulness**. Object detection supports all data feature metrics. [Supported Data Feature Metrics](#) provides all data feature metrics supported by ModelArts.

Data Feature Analysis

1. Log in to the [ModelArts management console](#). In the navigation pane on the left, choose **Asset Management > Datasets**.

2. Locate the target dataset, click **More** in the **Operation** column, and select **View Data Feature**. The **View Data Feature** tab of the dataset is displayed. You can also click a dataset name to go to the dataset page and click the **View Data Feature** tab.
3. By default, feature analysis is not started for published datasets. You need to manually start feature analysis tasks for each dataset version. On the **View Data Feature** tab, click **Analyze Features**.
4. In the displayed dialog box, configure the dataset version for feature analysis and click **Yes** to start analysis.

Version: Select a published version of the dataset.

Figure 6-61 Starting a data feature analysis task

Analyze Features



5. After a data feature analysis task is started, it takes a certain period of time to complete, depending on the data volume. If the selected version is displayed in the **Version** drop-down list and can be selected, the analysis is complete.
6. View the data feature analysis result.

Version: Select the version to be compared from the drop-down list. You can also select only one version.

Type: Select the type to be analyzed. The value can be **all**, **train**, **eval**, or **inference**.

Data Feature Metric: Select metrics to be displayed from the drop-down list. For details, see [Supported Data Feature Metrics](#).

Then, the selected version and metrics are displayed on the page. The displayed chart helps you understand data distribution for better data processing.
7. View historical records of the analysis task.

After data feature analysis is complete, you can click **Task History** on the right of the **Data Features** tab page to view historical analysis tasks and their statuses in the dialog box that is displayed.

Supported Data Feature Metrics

Table 6-35 Data feature metrics

Metric	Description	Explanation
Resolution	Image resolution. An area value is used as a statistical value.	Metric analysis results are used to check whether there is an offset point. If an offset point exists, you can resize or delete the offset point.
Aspect Ratio	An aspect ratio is a proportional relationship between an image's width and height.	The chart of the metric is in normal distribution, which is generally used to compare the difference between the training set and the dataset used in the real scenario.
Brightness	Brightness is the perception elicited by the luminance of a visual target. A larger value indicates better image brightness.	The chart of the metric is in normal distribution. You can determine whether the brightness of the entire dataset is high or low based on the distribution center. You can adjust the brightness based on your application scenario. For example, if the application scenario is night, the brightness should be lower.
Saturation	Color saturation of an image. A larger value indicates that the entire image color is easier to distinguish.	The chart of the metric is in normal distribution, which is generally used to compare the difference between the training set and the dataset used in the real scenario.
Clarity	Image clarity, which is calculated using the Laplace operator. A larger value indicates clearer edges and higher clarity.	You can determine whether the clarity meets the requirements based on the application scenario. For example, if data is collected from HD cameras, the clarity must be higher. You can sharpen or blur the dataset and add noises to adjust the clarity.

Metric	Description	Explanation
Colorfulness	Horizontal coordinate: Colorfulness of an image. A larger value indicates richer colors. Vertical coordinate: Number of images	Colorfulness on the visual sense, which is generally used to compare the difference between the training set and the dataset used in the real scenario.
Bounding Box Quantity	Horizontal coordinate: Number of bounding boxes in an image Vertical coordinate: Number of images	It is difficult for a model to detect a large number of bounding boxes in an image. Therefore, more images containing many bounding boxes are required for training.
Standard Deviation of Bounding Boxes Per Image	Horizontal coordinate: Standard deviation of bounding boxes in an image. If an image has only one bounding box, the standard deviation is 0. A larger standard deviation indicates higher bounding box size variation in an image. Vertical coordinate: Number of images	It is difficult for a model to detect a large number of bounding boxes with different sizes in an image. You can add data for training based on scenarios or delete data if such scenarios do not exist.
Aspect Ratio of Bounding Boxes	Horizontal coordinate: Aspect ratio of the target bounding boxes Vertical coordinate: Number of bounding boxes in all images	The chart of the metric is generally in Poisson distribution, which is closely related to application scenarios. This metric is mainly used to compare the differences between the training set and the validation set. For example, if the training set is a rectangle, the result will be significantly affected if the validation set is close to a square.

Metric	Description	Explanation
Area Ratio of Bounding Boxes	Horizontal coordinate: Area ratio of the target bounding boxes, that is, the ratio of the bounding box area to the entire image area. A larger value indicates a higher ratio of the object in the image. Vertical coordinate: Number of bounding boxes in all images	The metric is used to determine the distribution of anchors used in the model. If the target bounding box is large, set the anchor to a large value.
Marginalization Value of Bounding Boxes	Horizontal coordinate: Marginalization degree, that is, the ratio of the distance between the center point of the target bounding box and the center point of the image to the total distance of the image. A larger value indicates that the object is closer to the edge. (The total distance of an image is the distance from the intersection point of a ray (that starts from the center point of the image and passes through the center point of the bounding box) and the image border to the center point of the image.) Vertical coordinate: Number of bounding boxes in all images	Generally, the chart of the metric is in normal distribution. The metric is used to determine whether an object is at the edge of an image. If a part of an object is at the edge of an image, you can add a dataset or do not label the object.

Metric	Description	Explanation
Overlap Score of Bounding Boxes	Horizontal coordinate: Overlap degree, that is, the part of a single bounding box overlapped by other bounding boxes. The value ranges from 0 to 1. A larger value indicates that more parts are overlapped by other bounding boxes. Vertical coordinate: Number of bounding boxes in all images	The metric is used to determine the overlapping degree of objects to be detected. Overlapped objects are difficult to detect. You can add a dataset or do not label some objects based on your needs.
Brightness of Bounding Boxes	Horizontal coordinate: Brightness of the image in the target bounding box. A larger value indicates brighter image. Vertical coordinate: Number of bounding boxes in all images	Generally, the chart of the metric is in normal distribution. The metric is used to determine the brightness of an object to be detected. In some special scenarios, the brightness of an object is low and may not meet the requirements.
Clarity of Bounding Boxes	Horizontal coordinate: Clarity of the image in the target bounding box. A larger value indicates higher image clarity. Vertical coordinate: Number of bounding boxes in all images	The metric is used to determine whether the object to be detected is blurred. For example, a moving object may become blurred during collection and its data needs to be collected again.

6.7 Exporting Data from a ModelArts Dataset

6.7.1 Exporting Data from ModelArts to OBS

You can select certain data or obtain the required data by setting the filtering criteria. The data can be saved in OBS for future export. In this way, you can create a dataset. The historical export records can be viewed in task history.

This function is supported for only datasets of image classification, object detection, and image segmentation.

- For image classification datasets, only the label files in TXT format can be exported.
- For object detection datasets, only XML label files in Pascal VOC format can be exported.

- For image segmentation datasets, only XML label files in Pascal VOC format and mask images can be exported.

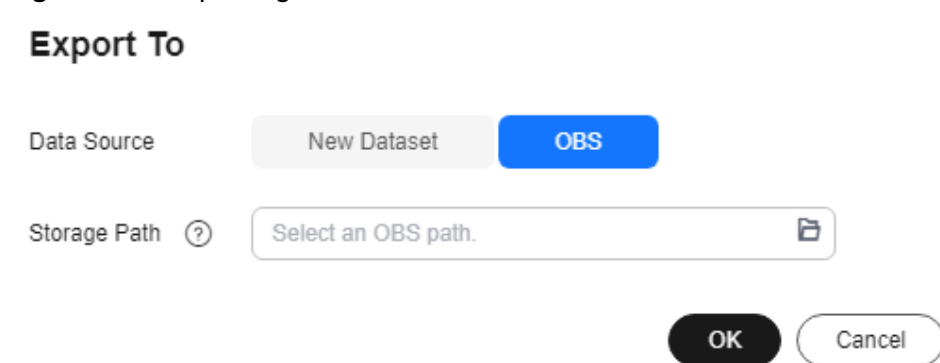
Exporting Data to OBS

1. Log in to the [ModelArts management console](#). In the navigation pane on the left, choose **Asset Management** > **Datasets**.
2. In the dataset list, select an image dataset and click the dataset name to go to the **Dashboard** tab of the dataset.
3. Click **Export** in the upper right corner. In the displayed **Export To** dialog box, enter the related information and click **OK**.

Data Source: Select **OBS**.

Storage Path: path where the data to be exported is stored. You are advised not to save data to the input or output path of the current dataset.

Figure 6-62 Exporting data to OBS



4. After the data is exported, view it in the specified path.
5. In the **Dashboard** tab, click **Export History** in the upper right corner. In the displayed dialog box, view the task history of the dataset.

Figure 6-63 Viewing the task history

View Task History

Task ID	Created	Type	Path	Total	Status
TWbANfmTN7z8zbyo4b	Apr 18, 2024 17:27:50 GMT+08:00	OBS	/modelarts-xy/test/	249	Successful

6.7.2 Exporting Data as a New Dataset

You can select certain data or obtain the required data by setting the filtering criteria to create a dataset. The historical export records can be viewed in task history. This section introduces how to export ModelArts data as a new dataset.

This function is supported for only datasets of image classification, object detection, and image segmentation.

- For image classification datasets, only the label files in TXT format can be exported.
- For object detection datasets, only XML label files in Pascal VOC format can be exported.
- For image segmentation datasets, only XML label files in Pascal VOC format and mask images can be exported.

Exporting Data to a New Dataset

1. Log in to the [ModelArts management console](#). In the navigation pane on the left, choose **Asset Management > Datasets**.
2. In the dataset list, select an image dataset and click the dataset name to go to the **Dashboard** tab of the dataset.
3. Click **Export** in the upper right corner. In the displayed **Export To** dialog box, enter the related information and click **OK**.

Data Source: Select **New Dataset**.

Name: name of the new dataset

Storage Path: input path of the new dataset, that is, the OBS path where the data to be exported is stored

Output Path: output path of the new dataset, that is, the output path after labeling is complete. The output path cannot be the same as the storage path, and the output path cannot be a subdirectory of the storage path.

Figure 6-64 Exporting to a new dataset

Export To

The 'Export To' dialog box contains the following elements:

- Data Source:** Two buttons, 'New Dataset' (highlighted in blue) and 'OBS'.
- Name:** A text input field with the placeholder text 'Enter a dataset name.'
- Storage Path:** A text input field with the placeholder text 'Select an OBS path.' and a folder icon on the right.
- Output Path:** A text input field with the placeholder text 'Select an OBS path.' and a folder icon on the right.
- Buttons:** 'OK' and 'Cancel' buttons at the bottom right.

4. After the data is exported, view it in the specified path. After the data is exported, you can view the new dataset in the dataset list.
5. In the **Dashboard** tab, click **Export History** in the upper right corner. In the displayed dialog box, view the task history of the dataset.

6.8 Getting Started: Creating an Object Detection Dataset

This section uses preparing data for training an object detection model as an example to describe how to analyze and label sample data. During actual service development, you can select one or more data management functions to prepare data based on service requirements. The operation process is as follows:

- [Making Preparations](#)
- [Creating a Dataset](#)

- [Analyzing Data](#)
- [Labeling Data](#)
- [Publishing Data](#)
- [Exporting Data](#)

 NOTE

Data management is being upgraded and is invisible to users who have not used data management.

Preparations

Before using data management of ModelArts, complete the following preparations:

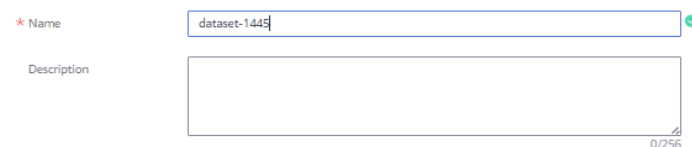
When using data management, ModelArts needs to access dependent services such as OBS. Therefore, grant permissions on the **Global Configuration** page. For details, see [Configuring Agency Authorization \(Recommended\)](#).

Creating a Dataset

In this example, an OBS path is used as the input path to create a dataset. Perform the following operations to create an object detection dataset and import the data to the dataset:

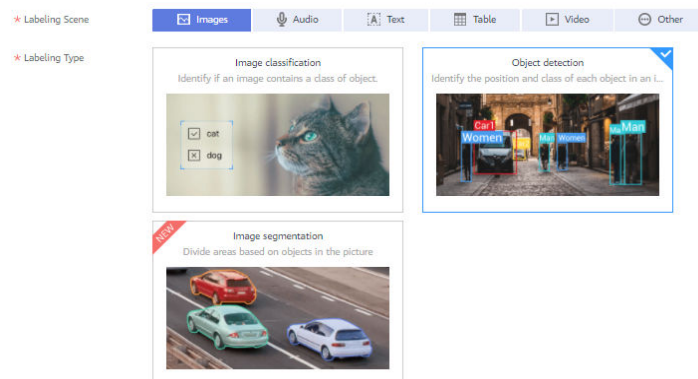
- Step 1** Log in to the [ModelArts management console](#). In the navigation pane, choose **Asset Management > Datasets**.
- Step 2** Click **Create**. On the **Create Dataset** page, create a dataset based on the data type and data labeling requirements.
1. Set the basic information, the name and description of the dataset.

Figure 6-65 Basic information of a dataset



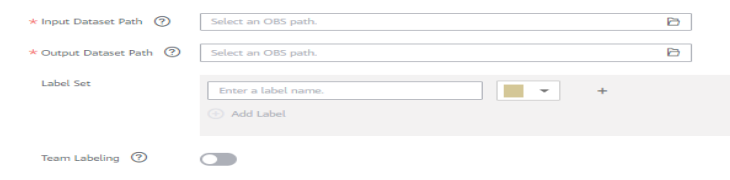
2. Set labeling scene and type. In this example, choose **Images** and **Object detection**.

Figure 6-66 Dataset labeling scene and type



3. Select an OBS path as **Input Dataset Path**, and select another OBS path as **Output Dataset Path**.

Figure 6-67 Input and output dataset path



4. After setting the parameters, click **Create** in the lower right corner to create a dataset.

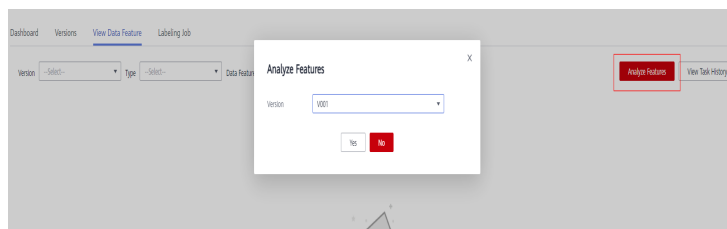
----End

Analyzing Data

After a dataset is created, you can perform data analysis based on image features, such as blurs and brightness, to better understand the data quality and determine whether the dataset meets your algorithm and model requirements.

1. Create a feature analysis task.
 - a. Before performing feature analysis, publish a dataset version. On the dataset **Dashboard** page, click **Publish** in the upper right corner to publish a new version of the dataset.
 - b. After the version is published, go to the **Dashboard** page. Click **View Data Feature** and **Feature Analysis**. In the displayed dialog box, select the newly published dataset version and click **OK** to start feature analysis.

Figure 6-68 Starting feature analysis



- c. View the task progress.

You can click **View Task History** to view the task progress. When the task status changes to **Successful**, the task is complete.

Figure 6-69 Feature analysis progress

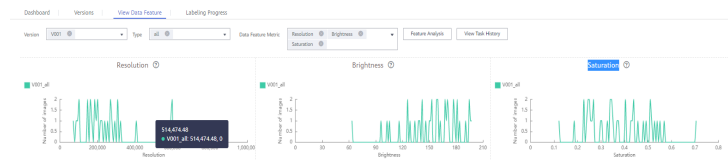
Dataset Version	Task ID	Created	Duration(hh...	Status
V002	Zlr07W8735...	Mar 16, 2022 ...	00:01:23	Successful

2. View feature analysis results.

After feature analysis is complete, you can select **Version**, **Type**, and **Data Feature Metric** on the **View Data Feature** tab page. Then, the selected versions and metrics are displayed on the page. The displayed chart helps you understand data distribution for a better understanding of your data.

- **Version:** Select one or more versions for comparison.
- **Type:** Select types to be analyzed. The values **all**, **train**, **eval**, and **inference** are available for you to select. They indicate all, training, evaluation, and inference, respectively.
- **Data Feature Metric:** Select the metrics to be displayed. For details about the metrics, see [Data feature metrics](#).

Figure 6-70 Viewing feature analysis results



In feature analysis results, for example, image brightness distribution is uneven, which means images of a certain brightness are lacking. This greatly affects model training. In this case, increase images of that brightness to make data more even for subsequent model building.

Labeling Data

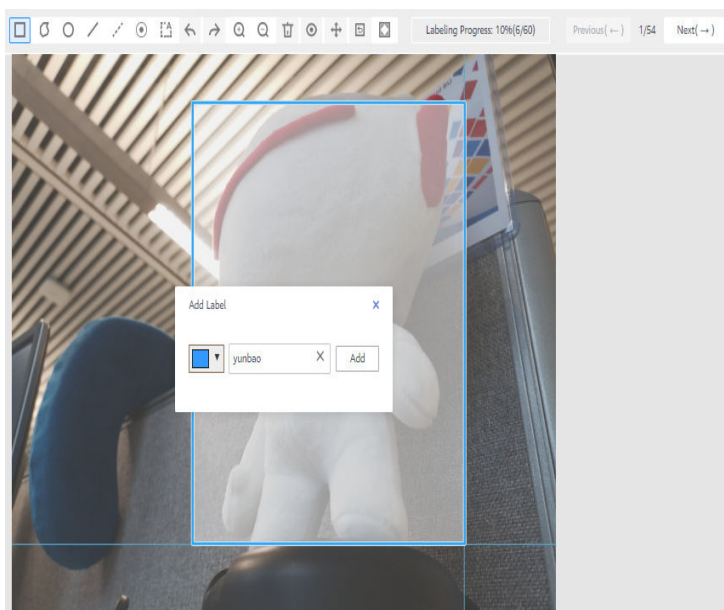
- Manual labeling
 - a. On the **Unlabeled** tab page, click an image. The system automatically directs you to the page for labeling the image.
 - b. On the toolbar of the labeling page, select a proper labeling tool. In this example, a rectangle is used for labeling.

Figure 6-71 Labeling tools



- c. Drag the mouse to select an object, enter a new label name in the displayed text box. If labels already exist, select one from the drop-down list box. Click **Add**.

Figure 6-72 Adding an object detection label



- d. Click **Back to Data Labeling Preview** in the upper left part of the page to view the labeling information. In the dialog box that is displayed, click **Yes** to save the labeling settings. The selected image is automatically moved to the **Labeled** tab page. On the **Unlabeled** and **All** tab pages, the labeling information is updated along with the labeling process, including the added label names and the number of images for each label.
- Auto labeling
Auto labeling allows you to automatically label remaining data after a small amount of data is manually labeled.
 - a. On the dataset details page, click **Auto Label** in the upper right corner.
 - b. In the **Enable Auto Labeling** dialog box, set the following parameters and click **Submit**.
 - **Auto Labeling Type: Active learning**
 - **Algorithm Type: Fast**
 Retain the default values of other parameters.

Figure 6-73 Starting auto labeling

Enable Auto Labeling

Auto Labeling Type ☒ Active learning ☐ Pre-labeling
The system uses semi-supervised learning and hard example filtering to perform auto labeling, reducing manual labeling workload and helping you find hard examples.

Algorithm Type ☒ Fast ☐ Precise

★ Specifications GPU: 1*NVIDIA-V100(32GB) | CPU: 8 vCPUs ...

⚠ To enable Auto Labeling, add at least one class of label to the data and ensure that the number of samples with the label is not less than 5. Auto Labeling can identify and add only rectangle labeling boxes.

Limited-time free
Auto Labeling is billed based on the training duration. [Pricing details](#)

Submit

- c. View auto labeling progress.

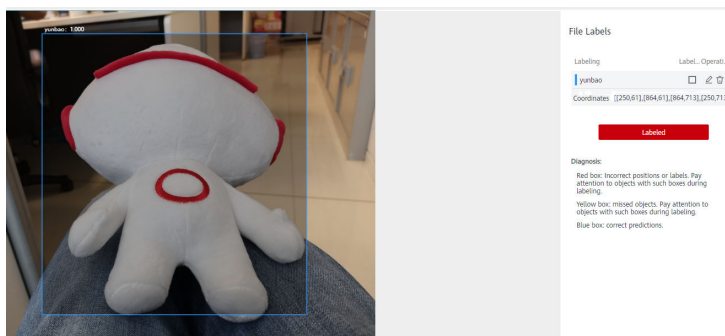
After auto labeling is started, you can view the task progress on the **To Be Confirmed** tab page. After a task is complete, you can view the automatically labeled data on the **To Be Confirmed** tab page.

Figure 6-74 Viewing auto labeling progress

- d. Confirm auto labeling results.

After auto labeling is complete, click the image on the **To be confirmed** tab page. On the labeling details page, you can view or modify the auto labeling result.

For correct labeling, click **Labeled** on the right. For wrong labeling, correct wrong labels. For auto labeling of object detection datasets, confirm images one by one. Ensure that all images are confirmed and go to the next step.

Figure 6-75 Confirming auto labeling results

Publishing Data

ModelArts training management allows you to create training jobs using ModelArts datasets or files in an OBS directory. If a dataset is used as the data source of a training job, specify a dataset and version. Therefore, you must have published a dataset version. For details, see [Labeling Images](#).

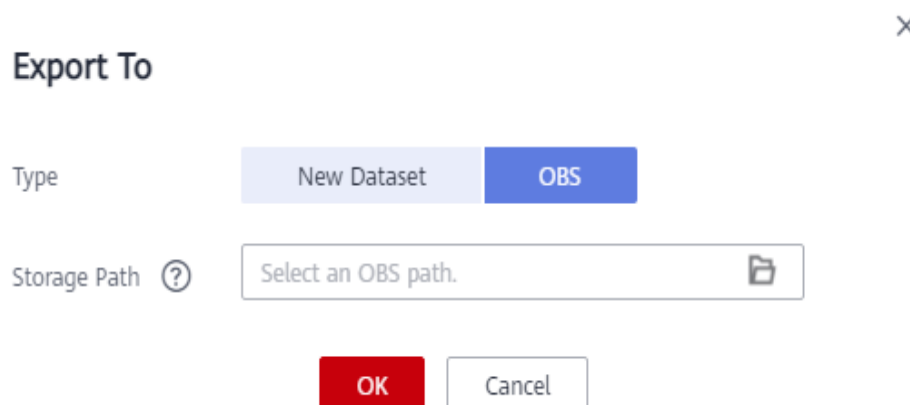
NOTE

Data that is from the same source and labeled in different batches are differentiated by version. This facilitates subsequent model building and development. You can select specified versions.

Exporting Data

ModelArts training management allows you to create training jobs using ModelArts datasets or files in an OBS directory. If you create a training job using an OBS directory, export the prepared data to OBS.

1. Export data to OBS.
 - a. On the dataset details page, select or filter the data to be exported, and click **Export** in the upper right corner.
 - b. Set **Type** to **OBS**, enter related information, and click **OK**.
Storage Path: path where the data to be exported is stored. You are advised not to save data to the input or output path of the current dataset.

Figure 6-76 Exporting to OBS

- c. After the data is exported, view it in the specified path.
2. View task history.

After exporting data, you can view the export task details in **Export History**.

 - a. On the dataset details page, click **Export History** in the upper right corner.
 - b. In the **View Task History** dialog box, view the export task history of the current dataset. You can view the task ID, creation time, export type, export path, total number of exported samples, and export status.

Figure 6-77 Export history

Task ID	Type	Path	Total	Status	Created
 No data available					

7 Using ModelArts Standard to Train Models

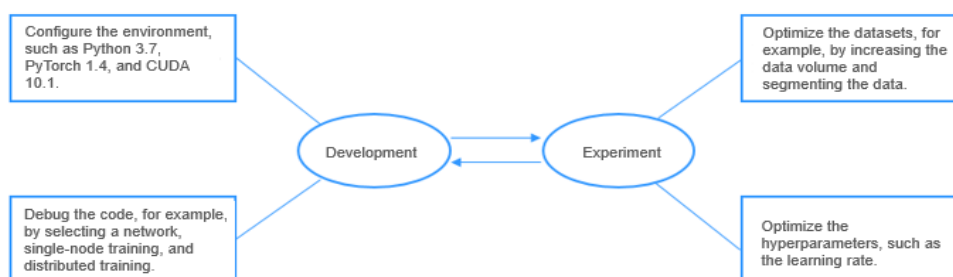
7.1 Model Training Process

AI modeling involves two stages:

- Development: Prepare and configure the environment, and debug code for training based on deep learning. ModelArts DevEnviron is recommended for code debugging.
- Experiment: Optimize the datasets and hyperparameters, and obtain an ideal model through multiple rounds of experiments. The ModelArts training platform is recommended for training.

In the two stages, code is designed, developed and tested in repeated cycles. In the development stage, when the code becomes stable, the modeling process enters the experiment stage, during which hyperparameters are continuously optimized to iterate the model. In the experiment stage, when the training performance can be optimized, the modeling process returns to the development stage for optimizing code.

Figure 7-1 Model development process



ModelArts provides model training, which allows you to view training results and tune model parameters based on the training results. You can select resource pools with different instance flavors for model training.

To train a model on ModelArts Standard, follow these steps:

Figure 7-2 ModelArts Standard model training process

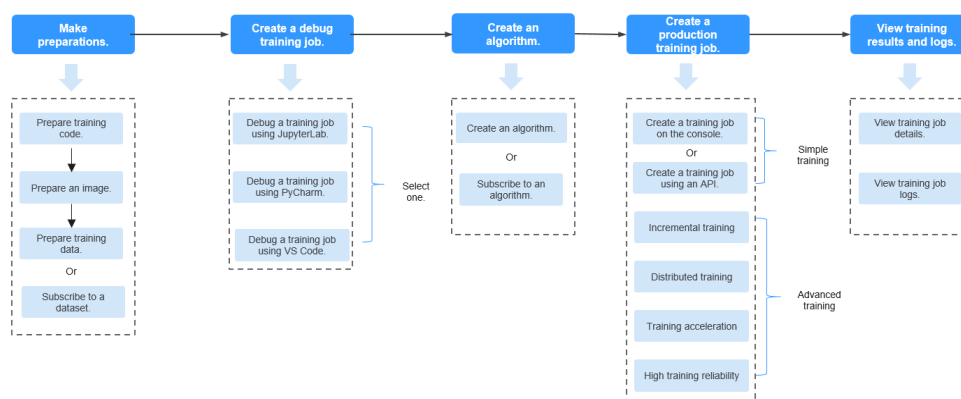


Table 7-1 ModelArts Standard model training process

Task	Subtask	Description
Making preparations	Preparing training code	Model training includes training code, training framework, and training data. Training code contains the boot file or command and dependency package of a training job. - To use a preset image to create a training job, develop training code by referring to Developing Code for Training Using a Preset Image. - To use a custom image to create a training job, develop training code by referring to Developing Code for Training Using a Custom Image.
	Preparing a training image	There are multiple training image sources. For details, see Preparing a Model Training Image . - ModelArts Standard offers mainstream preset images for model training, ready for immediate use. - If the preset images do not meet your needs, create custom images.
	Preparing training data	Before training, prepare necessary data, which can be datasets or predictive models. - Upload your training data to OBS if it does not need further processing. To create a training job, enter the OBS bucket path directly as the input parameter path. - Import your unlabeled or unprocessed training dataset to ModelArts data management for processing. To create a training job, choose your dataset in data management as the input parameter.

Task	Subtask	Description
Creating an algorithm	Creating an algorithm	Before creating a production training job, create an algorithm or subscribe to an algorithm from AI Gallery.
Creating a production training job	Using basic training features	- You can create a training job on the ModelArts Standard console. There are multiple algorithm types and training frameworks for you to select. For details, see Table 7-2. - ModelArts Standard also allows you to create training jobs using APIs. For details, see Using PyTorch to Create a Training Job (New-Version Training).
	Using advanced training features	ModelArts Standard supports the following advanced training features: - Incremental learning - Distributed training - Training acceleration - High training reliability
Viewing training results and logs	Viewing training job details	You can view a training job's parameter settings and events on the job details page at any time, whether the job is running or has completed.
	Viewing training job logs	Training logs track the execution and any errors that occur during training job runs. You can view these logs to identify and troubleshoot issues that cause jobs to fail.

Table 7-2 Methods of creating a training job

Creation Method	Description
Using a preset image to create a training job	If you have used some mainstream images to develop algorithms locally, you can select a mainstream image and create a training job to build a model.
Using a custom image to create a training job	To use a non-mainstream image, create a custom algorithm image and then use it to create a training job.
Using an existing algorithm to create a training job	In Algorithm Management, you can manage your created algorithms and those subscribed from AI Gallery. This allows you to quickly create training jobs and build models using these algorithms.

Creation Method	Description
Using a subscribed algorithm to create a training job	You can subscribe to algorithms in AI Gallery to quickly create training jobs and build models.

7.2 Preparing Model Training Code

7.2.1 Starting Training Using a Preset Image's Boot File

ModelArts Standard offers preset popular AI images tailored to their specific features. To train models efficiently, modify the boot command according to your chosen image's requirements for seamless execution.

ModelArts provides four preset images for creating training jobs.

Table 7-3 Preset images

Scenario	Preset Image	Description		Advantages and Suggestions	
NPU	Ascend-Powered-Engine	It is a set of AI images, runtime environments, and boot modes tailored for Huawei Cloud's AI accelerator chips. There are three boot modes:	Method 1: Using an RTF file to start a training job	Select this option if you use Huawei Cloud NPUs. This option can improve the model training efficiency.	RTF: general scenarios
			Method 2: Using the torchrun command to start a training job		torchrun: Use PyTorch +Ascend Extension for PyTorch.

Scenario	Preset Image	Description		Advantages and Suggestions	
			Method 3: Using the mspun command to start a training job		msrun: You can use this mode after MindSpore is installed. This option works independently of external libraries or configuration files, includes disaster recovery, and maintains high security.
GPU	PyTorch-GPU	PyTorch-GPU preset image.		This image features flexibility and ease of use, and works well for research and quick development.	
	TensorFlow-GPU	TensorFlow-GPU preset image.		This image excels in deploying models in production environments and optimizing performance, making it ideal for enterprise applications.	
	Horovod/MPI/MindSpore-GPU	ModelArts uses mpirun to run boot files for Horovod, MPI, or MindSpore-GPU.		This feature requires the OpenMPI library. Choose it only if you are familiar with OpenMPI.	

This section describes how to modify the boot file when creating a training job using different preset images.

Ascend-Powered-Engine

Ascend-Powered-Engine is a unique engine that combines an AI framework, runtime environment, and boot mode tailored to Ascend accelerator chips. Unlike conventional AI frameworks like PyTorch or TensorFlow, or parallel execution frameworks such as MPI, it serves a distinct purpose.

Snt9 Ascend accelerators run on Arm CPU environments, which means their Docker images are Arm images. Ascend-Powered-Engine images include the Huawei's CANN (heterogeneous computing architecture) compute library instead of the CUDA (unified computing architecture) compute library used in GPU setups. CANN supports AI tasks and works with Ascend drivers.

After a training job is submitted, ModelArts Standard automatically runs the boot file. When using the Ascend-Powered-Engine framework, both single-node and distributed jobs start with the same command.

The Ascend-Powered-Engine framework offers three boot modes. By default, the boot file is executed based on **RANK_TABLE_FILE**. You can also configure the **MA_RUN_METHOD** environment variable to run the boot file using alternative methods. The **MA_RUN_METHOD** environment variable offers two boot options: **torchrun** and **msrun**.

- **Method 1: Using an RTF file to start a training job**

If the environment variable **MA_RUN_METHOD** is not configured, ModelArts Standard uses a rank table file to start the boot file of the training job by default.

The number of times the boot file runs for each training job depends on the number of PUs used. When a job is running, the boot file is executed once for each PU. For example, in a single-node job with one PU, the boot file runs once. In a single-node job with eight PUs, it runs eight times. So, do not listen on ports in the boot file.

The following environment variables are automatically configured in the boot file:

- **RANK_TABLE_FILE**: rank table file path.
- **ASCEND_DEVICE_ID**: logical device ID. For example, for single-PU training, the value is always **0**.
- **RANK_ID**: logical (sequential) number of a device in a training job.
- **RANK_SIZE**: Set this parameter based on the number of devices in the rank table file. For example, the value is **4** for 4 Snt9b devices.

To ensure the boot file runs only once, check the **ASCEND_DEVICE_ID** value. If it is **0**, execute the logic; otherwise, exit directly.

To enable ranktable dynamic routing for training network acceleration, add the environment variable **ROUTE_PLAN=true**. For details, see [Enabling Dynamic Route Acceleration for Training Jobs](#).

For details about the example code file **mindspore-verification.py** of the Ascend-Powered-Engine framework, see [Training the mindspore-verification.py File](#).

- **Method 2: Using the torchrun command to start a training job**

If the environment variable **MA_RUN_METHOD** is set **torchrun**, ModelArts Standard uses the **torchrun** command to run the boot file.

 NOTE

The PyTorch version must be 1.11.0 or later.

- For single-node jobs, ModelArts Standard uses these commands to start the boot file:

```
torchrun --standalone --nnodes=${MA_NUM_HOSTS} --nproc_per_node=${MA_NUM_GPUS} $  
{MA_EXTRA_TORCHRUN_PARAMS} "Boot file" {arg1} {arg2} ...
```

- For multi-node jobs, ModelArts Standard uses these commands to start the boot file:

```
torchrun --nnodes=${MA_NUM_HOSTS} --nproc_per_node=${MA_NUM_GPUS} --node_rank=$  
{VC_TASK_INDEX} --master_addr={master_addr} --master_port=$  
{MA_TORCHRUN_MASTER_PORT} --rdzv_id={ma_job_name} --rdzv_backend=static $  
{MA_EXTRA_TORCHRUN_PARAMS} "Boot file" {arg1} {arg2} ...
```

Parameters:

Table 7-4 Parameters for starting a training job using the **torchrun** command

Parameter	Description
standalone	Identifier of a single-node job.
nnodes	Number of task nodes.
nproc_per_node	Number of main processes started by each task node. Set this parameter to the number of NPUs allocated to the task.
node_rank	Task rank, which is used for multi-task distributed training.
master_addr	Address of the main task (rank 0). Set it to the communication domain name of worker-0 .
master_port	Port used for communication during distributed training on the main task (rank 0). The default value is 18888 . When a master_port conflict occurs, you can modify the port configuration by configuring the MA_TORCHRUN_MASTER_PORT environment variable.
rdzv_id	Rendezvous ID. Set it to a value with the training job ID.
rdzv_backend	Rendezvous backend, which is fixed at static . That is, master_addr and master_port are used instead of Rendezvous.

- Additionally, you can configure the **MA_EXTRA_TORCHRUN_PARAMS** environment variable to add additional **torchrun** command parameters or overwrite the preset **torchrun** command parameters. The following is an example of configuring the **rdzv_conf** parameter in the **torchrun** command:

```
"environments": {  
  "MA_RUN_METHOD": "torchrun",  
  "MA_EXTRA_TORCHRUN_PARAMS": "--rdzv_conf=timeout=7200"  
}
```

If the **RuntimeError: Socket Timeout** error occurs during the distributed process group initialization using **torchrun**, go to [1](#) and further locate the fault.

- **Method 3: Using the mrun command to start a training job**

If the environment variable **MA_RUN_METHOD** is set **msrun**, ModelArts Standard uses the **msrun** command to run the boot file.

 NOTE

The MindSpore version must be 2.3.0 or later.

This solution supports dynamic networking and rank table file-based networking. If you set the environment variable **MS_RANKTABLE_ENABLE** to **True**, **msrun** reads the rank table file for networking. Otherwise, dynamic networking is used by default.

msrun uses these commands to start the boot file:

```
msrun --worker_num=${msrun_worker_num} --local_worker_num=${MA_NUM_GPUS} --master_addr=${msrun_master_addr} --node_rank=${VC_TASK_INDEX} --master_port=${msrun_master_port} --log_dir=${msrun_log_dir} --join=True --cluster_time_out=${MSRUN_CLUSTER_TIME_OUT} --rank_table_file=${msrun_rank_table_file} "Boot file" {arg1} {arg2} ...
```

Parameters:

Table 7-5 Parameters for starting a training job using the **msrun** command

Parameter	Description
worker_num	Total number of processes, which is also equivalent to the number of PUs involved, as each PU initiates a process.
local_worker_num	Number of processes on the current node, which is also the number of PUs used by the current node.
master_addr	IP address of the node where the msrun scheduling process is located. This parameter does not need to be configured for single-node jobs.
master_port	Port number of the msrun scheduling process.
node_rank	Port number of the msrun scheduling process.
log_dir	Log output directory of msrun and all processes.
join	Specifies whether the msrun process still exists after the training process is started. The default value is True , indicating that the msrun process exits after all processes exit.
cluster_time_out	Timeout interval of the cluster networking. The default value is 600s . The value can be controlled by the MSRUN_CLUSTER_TIME_OUT environment variable.
rank_table_file	Address of the rank table file. If the environment variable MS_RANKTABLE_ENABLE is set to True , this parameter is added during startup.

PyTorch-GPU

For single-node multi-PU scenarios, the platform adds the **--init_method "tcp://<ip>:<port>"** parameter to the boot file.

For multi-node multi-PU scenarios, the platform adds the **--init_method "tcp://<ip>:<port>" --rank <rank_id> --world_size <node_num>** parameter to the boot file.

The preceding parameters must be parsed in the boot file.

For details about the code example of the PyTorch-GPU framework, see "Method 1" in [Example: Creating a DDP Distributed Training Job \(PyTorch + GPU\)](#).

TensorFlow-GPU

For a single-node job, ModelArts starts a training container that exclusively uses the resources on the node.

For a multi-node job, ModelArts starts a parameter server and a worker on the same node. It allocates parameter server and worker tasks in a 1:1 ratio. For example, in a two-node job, two parameter servers and two workers are allocated. ModelArts also injects the following parameters into the boot file:

```
--task_index <VC_TASK_INDEX> --ps_hosts <TF_PS_HOSTS> --worker_hosts <TF_WORKER_HOSTS> --  
job_name <MA_TASK_NAME>
```

The following parameters must be parsed in the boot file.

Table 7-6 Parameters

Parameter	Description
VC_TASK_INDEX	Task serial number, for example, 0 , 1 , or 2 .
TF_PS_HOSTS	Addresses of parameter server nodes, for example, [xx-ps-0.xx:TCP_PORT,xx-ps-1.xx:TCP_PORT]. The value of <i>TCP_PORT</i> is a random port ranging from 5,000 to 10,000.
TF_WORKER_HOSTS	Addresses of worker nodes, for example, [xx-worker-0.xx:TCP_PORT,xx-worker-1.xx:TCP_PORT]. The value of <i>TCP_PORT</i> is a random port ranging from 5,000 to 10,000.
MA_TASK_NAME	Task name, which can be ps or worker .

Horovod/MPI/MindSpore-GPU

ModelArts uses **mpirun** to run training boot files for Horovod, MPI, or MindSpore-GPU. To use a preset engine in ModelArts Standard, simply edit the boot file (training script). ModelArts Standard automatically builds the **mpirun** command and training job cluster. The platform does not add extra parameters to the boot file.

Example of **pytorch_synthetic_benchmark.py**:

```
import argparse
import torch.backends.cudnn as cudnn
import torch.nn.functional as F
import torch.optim as optim
import torch.utils.data.distributed
from torchvision import models
import horovod.torch as hvd
import timeit
import numpy as np

# Benchmark settings
parser = argparse.ArgumentParser(description='PyTorch Synthetic Benchmark',
                                formatter_class=argparse.ArgumentDefaultsHelpFormatter)
parser.add_argument('--fp16-allreduce', action='store_true', default=False,
                    help='use fp16 compression during allreduce')

parser.add_argument('--model', type=str, default='resnet50',
```

```
        help='model to benchmark')
parser.add_argument('--batch-size', type=int, default=32,
                    help='input batch size')

parser.add_argument('--num-warmup-batches', type=int, default=10,
                    help='number of warm-up batches that don\'t count towards benchmark')
parser.add_argument('--num-batches-per-iter', type=int, default=10,
                    help='number of batches per benchmark iteration')
parser.add_argument('--num-iters', type=int, default=10,
                    help='number of benchmark iterations')

parser.add_argument('--no-cuda', action='store_true', default=False,
                    help='disables CUDA training')

parser.add_argument('--use-adasum', action='store_true', default=False,
                    help='use adasum algorithm to do reduction')

args = parser.parse_args()
args.cuda = not args.no_cuda and torch.cuda.is_available()

hvd.init()

if args.cuda:
    # Horovod: pin GPU to local rank.
    torch.cuda.set_device(hvd.local_rank())

cudnn.benchmark = True

# Set up standard model.
model = getattr(models, args.model)()

# By default, Adasum doesn't need scaling up learning rate.
lr_scaler = hvd.size() if not args.use_adasum else 1

if args.cuda:
    # Move model to GPU.
    model.cuda()
    # If using GPU Adasum allreduce, scale learning rate by local_size.
    if args.use_adasum and hvd.nccl_built():
        lr_scaler = hvd.local_size()

optimizer = optim.SGD(model.parameters(), lr=0.01 * lr_scaler)

# Horovod: (optional) compression algorithm.
compression = hvd.Compression.fp16 if args.fp16_allreduce else hvd.Compression.none

# Horovod: wrap optimizer with DistributedOptimizer.
optimizer = hvd.DistributedOptimizer(optimizer,
                                     named_parameters=model.named_parameters(),
                                     compression=compression,
                                     op=hvd.Adasum if args.use_adasum else hvd.Average)

# Horovod: broadcast parameters & optimizer state.
hvd.broadcast_parameters(model.state_dict(), root_rank=0)
hvd.broadcast_optimizer_state(optimizer, root_rank=0)

# Set up fixed fake data
data = torch.randn(args.batch_size, 3, 224, 224)
target = torch.LongTensor(args.batch_size).random_() % 1000
if args.cuda:
    data, target = data.cuda(), target.cuda()

def benchmark_step():
    optimizer.zero_grad()
    output = model(data)
    loss = F.cross_entropy(output, target)
    loss.backward()
    optimizer.step()
```

```
def log(s, nl=True):
    if hvd.rank() != 0:
        return
    print(s, end='\n' if nl else '')

log('Model: %s' % args.model)
log('Batch size: %d' % args.batch_size)
device = 'GPU' if args.cuda else 'CPU'
log('Number of %ss: %d' % (device, hvd.size()))

# Warm-up
log('Running warmup...')
timeit.timeit(benchmark_step, number=args.num_warmup_batches)

# Benchmark
log('Running benchmark...')
img_secs = []
for x in range(args.num_iters):
    time = timeit.timeit(benchmark_step, number=args.num_batches_per_iter)
    img_sec = args.batch_size * args.num_batches_per_iter / time
    log('Iter #%d: %.1f img/sec per %s' % (x, img_sec, device))
    img_secs.append(img_sec)

# Results
img_sec_mean = np.mean(img_secs)
img_sec_conf = 1.96 * np.std(img_secs)
log('Img/sec per %s: %.1f +-%.1f' % (device, img_sec_mean, img_sec_conf))
log('Total img/sec on %d %s(s): %.1f +-%.1f' %
    (hvd.size(), device, hvd.size() * img_sec_mean, hvd.size() * img_sec_conf))
```

run_mpi.sh is as follows:

```
#!/bin/bash
MY_HOME=/home/ma-user

MY_SSHD_PORT=${MY_SSHD_PORT:-"36666"}

MY_MPI_BTL_TCP_IF=${MY_MPI_BTL_TCP_IF:-"eth0,bond0"}

MY_TASK_INDEX=${MA_TASK_INDEX:-${VC_TASK_INDEX:-${VK_TASK_INDEX}}}

MY_MPI_SLOTS=${MY_MPI_SLOTS:-"${MA_NUM_GPUS}"}

MY_MPI_TUNE_FILE="${MY_HOME}/env_for_user_process"

if [ -z ${MY_MPI_SLOTS} ]; then
    echo "[run_mpi] MY_MPI_SLOTS is empty, set it be 1"
    MY_MPI_SLOTS="1"
fi

printf "MY_HOME: ${MY_HOME}\nMY_SSHD_PORT: ${MY_SSHD_PORT}\nMY_MPI_BTL_TCP_IF: ${MY_MPI_BTL_TCP_IF}\nMY_TASK_INDEX: ${MY_TASK_INDEX}\nMY_MPI_SLOTS: ${MY_MPI_SLOTS}\n"

env | grep -E '^MA_|^S3_|^PATH|^VC_WORKER|^SCC|^CRED' | grep -v '=' > $MY_MPI_TUNE_FILE
# add -x to each line
sed -i 's/^-x /' ${MY_MPI_TUNE_FILE}

sed -i "s|{{MY_SSHD_PORT}}|${MY_SSHD_PORT}|g" ${MY_HOME}/etc/ssh/sshd_config

# start sshd service
bash -c "$(which sshd) -f ${MY_HOME}/etc/ssh/sshd_config"

# confirm the sshd is up
netstat -anp | grep LIS | grep ${MY_SSHD_PORT}

if [ $MY_TASK_INDEX -eq 0 ]; then
```

```
# generate the hostfile of mpi
for ((i=0; i<${MA_NUM_HOSTS}; i++))
do
    eval hostname=${MA_VJ_NAME}-${MA_TASK_NAME}-${i}.${MA_VJ_NAME}
    echo "[run_mpi] hostname: ${hostname}"

    ip=""
    while [ -z "$ip" ]; do
        ip=$(ping -c 1 ${hostname} | grep "PING" | sed -E 's/PING .* .([0-9.]+). */\1/g')
        sleep 1
    done
    echo "[run_mpi] resolved ip: ${ip}"

    # test the sshd is up
    while :
    do
        if [ cat < /dev/null >/dev/tcp/${ip}/${MY_SSHD_PORT} ]; then
            break
        fi
        sleep 1
    done

    echo "[run_mpi] the sshd of ip ${ip} is up"

    echo "${ip} slots=${MY_MPI_SLOTS}" >> ${MY_HOME}/hostfile
done

printf "[run_mpi] hostfile:\n`cat ${MY_HOME}/hostfile`\n"
fi

RET_CODE=0

if [ ${MY_TASK_INDEX} -eq 0 ]; then

    echo "[run_mpi] start exec command time: "$(date +"%Y-%m-%d-%H:%M:%S")

    np=$(( ${MA_NUM_HOSTS} * ${MY_MPI_SLOTS} ))

    echo "[run_mpi] command: mpirun -np ${np} -hostfile ${MY_HOME}/hostfile -mca plm_rsh_args \"-p $
${MY_SSHD_PORT}\" -tune ${MY_MPI_TUNE_FILE} ... @$"

    # execute mpirun at worker-0
    # mpirun
    mpirun \
        -np ${np} \
        -hostfile ${MY_HOME}/hostfile \
        -mca plm_rsh_args "-p ${MY_SSHD_PORT}" \
        -tune ${MY_MPI_TUNE_FILE} \
        -bind-to none -map-by slot \
        -x NCCL_DEBUG=INFO -x NCCL_SOCKET_IFNAME=${MY_MPI_BTL_TCP_IF} -x
NCCL_SOCKET_FAMILY=AF_INET \
        -x HOROVOD_MPI_THREADS_DISABLE=1 \
        -x LD_LIBRARY_PATH \
        -mca pml ob1 -mca btl ^openib -mca plm_rsh_no_tree_spawn true \
        "$@"

    RET_CODE=$?

    if [ $RET_CODE -ne 0 ]; then
        echo "[run_mpi] exec command failed, exited with $RET_CODE"
    else
        echo "[run_mpi] exec command successfully, exited with $RET_CODE"
    fi

    # stop 1...N worker by killing the sleep proc
    sed -i '1d' ${MY_HOME}/hostfile
    if [ `cat ${MY_HOME}/hostfile | wc -l` -ne 0 ]; then
        echo "[run_mpi] stop 1 to (N - 1) worker by killing the sleep proc"
```

```
sed -i 's/${MY_MPI_SLOTS}/1/g' ${MY_HOME}/hostfile
printf "[run_mpi] hostfile:\n`cat ${MY_HOME}/hostfile`\n"

mpirun \
--hostfile ${MY_HOME}/hostfile \
--mca btl_tcp_if_include ${MY_MPI_BTL_TCP_IF} \
--mca plm_rsh_args "-p ${MY_SSHD_PORT}" \
-x PATH -x LD_LIBRARY_PATH \
pkill sleep \
> /dev/null 2>&1
fi

echo "[run_mpi] exit time: "$(date +"%Y-%m-%d-%H:%M:%S")
else
echo "[run_mpi] the training log is in worker-0"
sleep 365d
echo "[run_mpi] exit time: "$(date +"%Y-%m-%d-%H:%M:%S")
fi

exit $RET_CODE
```

FAQs

1. What Should I Do If RuntimeError: Socket Timeout Is Displayed During Distributed Process Group Initialization Using torchrun?

If the **RuntimeError: Socket Timeout** error occurs during the distributed process group initialization using **torchrun**, you can add the following environment variables to create a training job again to view initialization details and further locate the fault.

- LOGLEVEL=INFO
- TORCH_CPP_LOG_LEVEL=INFO
- TORCH_DISTRIBUTED_DEBUG=DETAIL

The **RuntimeError: Socket Timeout** error is caused by a significant time discrepancy between tasks when running the **torchrun** command. The time discrepancy is caused by initialization tasks, like downloading the training data and checkpoint read/write, which happen before the **torchrun** command is run. If the time taken to complete these initialization tasks varies significantly, a Socket Timeout error may occur. When this error happens, check the time difference between the **torchrun** execution points for each task. If the time difference is too large, optimize the initialization process before running the **torchrun** command to ensure a reasonable time gap.

7.2.2 Developing Code for Training Using a Preset Image

On ModelArts Standard, you must use preset images for building and training models. If these images do not fit specific needs, customizing them becomes necessary. Before creating an algorithm with a preset image, writing algorithm code ensures it fully supports your requirements. This preparation improves efficiency in model training and optimization, streamlining the overall development process.

Configuration Path

When creating an algorithm, set the code directory, boot file, input path, and output path. These settings enable the interaction between your codes and ModelArts Standard.

- **Code directory**

Specify the code directory in the OBS bucket and upload training data such as training code, dependency installation packages, or pre-generated model to the directory. After you create a training job, ModelArts downloads the code directory and its subdirectories to the container.

Take OBS path **obs://obs-bucket/training-test/demo-code** as an example. The content in the OBS path will be automatically downloaded to **\$ {MA_JOB_DIR}/demo-code** in the training container, and **demo-code** (customizable) is the last-level directory of the OBS path.

Do not store training data in the code directory. When the training job starts, the data stored in the code directory will be downloaded to the backend. A large amount of training data may lead to a download failure. It is recommended that the size of the code directory does not exceed 50 MB.
- **Boot file**

The boot file in the code directory is used to start the training. Only Python boot files are supported. For details about the boot process of the boot file of a preset image, see [Starting Training Using a Preset Image's Boot File](#).
- **Input path**

The training data must be uploaded to an OBS bucket or stored in the **Dataset**. In the training code, the input path must be parsed. ModelArts automatically downloads the data in the input path to the local container directory for training. Ensure that you have the read permission to the OBS bucket. After the training job is started, ModelArts mounts a disk to the **/cache** directory. You can use this directory to store temporary files. For details about the size of the **/cache** directory, see [What Are Sizes of the /cache Directories for Different Resource Specifications in the Training Environment?](#)
- **Output path**

You are advised to set an empty directory as the training output path. In the training code, the output path must be parsed. ModelArts automatically uploads the training output to the output path. Ensure that you have the write and read permissions on the OBS bucket.

In ModelArts, the training code must contain the steps in [\(Optional\) Introducing Dependencies](#) and [Parsing and Setting Input and Output Paths](#).

(Optional) Introducing Dependencies

If your model references other dependencies, place the required file or installation package in **Code Directory** you set during algorithm creation.

- For details about how to install the Python dependency package, see [How Do I Create a Training Job When a Dependency Package Is Referenced by the Model to Be Trained?](#)
- For details about how to install a C++ dependency library, see [How Do I Install a Library That C++ Depends on?](#)
- For details about how to load parameters to a pre-trained model, see [How Do I Load Some Well Trained Parameters During Job Training?](#)

Parsing and Setting Input and Output Paths

When a ModelArts Standard training job reads data stored in OBS or outputs training results to a specified OBS path, perform the following operations to configure the input and output data:

1. Parse the input and output paths in the training code. The following method is recommended:

```
import argparse
# Create a parsing task.
parser = argparse.ArgumentParser(description='train mnist')

# Add parameters.
parser.add_argument('--data_url', type=str, default="./Data/mnist.npz", help='path where the dataset is saved')
parser.add_argument('--train_url', type=str, default="./Model", help='path where the model is saved')

# Parse the parameters.
args = parser.parse_args()
```

After the parameters are parsed, use **data_url** and **train_url** to replace the paths to the data source and the data output, respectively.

2. When creating a training job, set the input and output paths.

Select the OBS path or dataset path as the training input, and the OBS path as the output.

Complete Training Code Example

The training code is closely related to the AI engine you use. The following uses the TensorFlow framework as an example. Before using this case, you need to [download](#) the **mnist.npz** file and upload it to the OBS bucket. The training input is the OBS path where the **mnist.npz** file is stored.

The following training code example contains the code for saving the model.

```
import os
import argparse
import tensorflow as tf

parser = argparse.ArgumentParser(description='train mnist')
parser.add_argument('--data_url', type=str, default="./Data/mnist.npz", help='path where the dataset is saved')
parser.add_argument('--train_url', type=str, default="./Model", help='path where the model is saved')
args = parser.parse_args()

mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data(args.data_url)
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10)
])

loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
model.compile(optimizer='adam',
              loss=loss_fn,
              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=5)

model.save(os.path.join(args.train_url, 'model'))
```

7.2.3 Developing Code for Training Using a Custom Image

ModelArts Standard offers various preset images and algorithms for most needs. If these do not suit you, its custom image training feature allows flexibility. You can create a tailored image to design the exact training environment required.

Customizing an image requires a deep understanding of containers. Use this method only if the subscribed algorithms and preset images cannot meet your requirements. Custom images can be used to train models in ModelArts Standard only after they are uploaded to the Software Repository for Container (SWR).

Boot Command Specifications for Custom Images

You can create an image based on the ModelArts image specifications, select your own image and configure the code directory (optional) and boot command to create a training job.

Figure 7-3 Selecting a custom image

The screenshot shows the 'Selecting a custom image' interface. It features several sections:

- Algorithm Type:** Three tabs: 'Custom algorithm' (active), 'My algorithm', and 'My subscription'. A help icon (?) is on the right.
- Boot Mode:** Two buttons: 'Preset image' and 'Custom image' (active).
- Image:** A text input field and a 'Select' button.
- Code Directory:** A text input field and a 'Select' button.
- User ID:** A text input field containing '1000' and a help icon (?) on the left.
- Boot Command:** A text input field containing '1' and a help icon (?) on the left.

WARNING

1. When you use a custom image to create a training job, the boot command must be executed in the **/home/ma-user** directory. Otherwise, the training job may run abnormally.
2. If there are input pipes, output pipes, or hyperparameters, ensure that the last command in the boot commands runs the training script. Otherwise, an error is reported.

1. You must use **conda env** to start training jobs created using custom images. Training jobs do not run in a shell. Therefore, you are not allowed to run the **conda activate** command to activate a specified Conda environment. In this case, use other methods to start training. For example, Conda in your custom image is installed in the **/home/ma-user/anaconda3** directory, the Conda environment is **python-3.7.10**, and the training script is stored in **/home/ma-user/modelarts/user-job-dir/code/train.py**. Use a specified Conda environment to start training in one of the following ways:
 - Method 1: Configure the correct **DEFAULT_CONDA_ENV_NAME** and **ANACONDA_DIR** environment variables for the image.

```
ANACONDA_DIR=/home/ma-user/anaconda3  
DEFAULT_CONDA_ENV_NAME=python-3.7.10
```

Run the **python** command to start the training script. The following shows an example:

```
python /home/ma-user/modelarts/user-job-dir/code/train.py
```

- Method 2: Use the absolute path of Conda environment Python.

Run the **/home/ma-user/anaconda3/envs/python-3.7.10/bin/python** command to start the training script. The following shows an example:

```
/home/ma-user/anaconda3/envs/python-3.7.10/bin/python /home/ma-user/modelarts/user-job-dir/code/train.py
```

- Method 3: Configure the **PATH** environment variable.

Configure the bin directory of the specified Conda environment into the path environment variable. Run the **python** command to start the training script. The following shows an example:

```
export PATH=/home/ma-user/anaconda3/envs/python-3.7.10/bin:$PATH; python /home/ma-user/modelarts/user-job-dir/code/train.py
```

- Method 4: Run the **conda run -n** command.

Run the **/home/ma-user/anaconda3/bin/conda run -n python-3.7.10** command to execute the training. The following shows an example:

```
/home/ma-user/anaconda3/bin/conda run -n python-3.7.10 python /home/ma-user/modelarts/user-job-dir/code/train.py
```

2. The last command in the boot commands must run the training script.

If there are input pipes, output pipes, or hyperparameters, ensure that the last command in the boot commands runs the training script.

Reason: The system appends input pipes, output pipes, and hyperparameters to the end of the boot commands. If the last command is not the training script, an error will occur.

Example: If the last line of the boot commands is **python train.py** and the **--data_url** hyperparameter exists, the system executes **python train.py --data_url=/input** when running properly. However, if the boot commands end with another command, such as:

```
python train.py  
pwd # The last command is pwd instead of the training script.
```

The system will execute **python train.py pwd --data_url=/input**, leading to an error.

Training Code Adaptation Specifications for Training Using an Ascend-powered Custom Image

When creating a training job that uses NPU resources, the system automatically generates the **Ascend HCCL RANK_TABLE_FILE** file in the training container. When using a preset image, **Ascend HCCL RANK_TABLE_FILE** is automatically parsed during training. When using a custom image, the training code must be modified to read and parse **Ascend HCCL RANK_TABLE_FILE**.

Ascend HCCL RANK_TABLE_FILE file description

Ascend HCCL RANK_TABLE_FILE provides the cluster used by Ascend distributed training jobs. It is used for distributed communication between Ascend chips and can be parsed by Huawei Collective Communication Library (HCCL). The file has two format versions: template 1 and template 2.

- ModelArts provides the template 2 format. The **Ascend HCCL RANK_TABLE_FILE** file in the ModelArts training environment is named **jobstart_hccl.json**. You can access this file using the preset **RANK_TABLE_FILE** environment variable.

Table 7-7 RANK_TABLE_FILE environment variables

Environment Variable	Description
RANK_TABLE_FILE	Directory of Ascend HCCL RANK_TABLE_FILE , which is /user/config. Obtain the file using `\${RANK_TABLE_FILE}/jobstart_hccl.json` .

Example of the **jobstart_hccl.json** file content in the ModelArts training environment (template 2):

```
{
  "group_count": "1",
  "group_list": [{
    "device_count": "1",
    "group_name": "job-trainjob",
    "instance_count": "1",
    "instance_list": [{
      "devices": [{
        "device_id": "4",
        "device_ip": "192.1.10.254"
      }],
      "pod_name": "jobxxxxxxx-job-trainjob-0",
      "server_id": "192.168.0.25"
    }],
    "status": "completed"
  }
}
```

In **jobstart_hccl.json**, the **status** value may not be **completed** when the training script is started. In this case, wait until the **status** value changes to **completed** and read the remaining content of the file.

- After the **status** field is **completed**, use the training script to convert the **jobstart_hccl.json** file from template 2 to template 1 format.

Format of the **jobstart_hccl.json** file after format conversion (template 1):

```
{
  "server_count": "1",
  "server_list": [{
    "device": [{
      "device_id": "4",
      "device_ip": "192.1.10.254",
      "rank_id": "0"
    }],
    "server_id": "192.168.0.25"
  }],
  "status": "completed",
  "version": "1.0"
}
```

Mount Points of a Training Job in a Container

When training a model with a custom image, the mount points in the container are shown in [Table 7-8](#).

Table 7-8 Training job mount points

Mount Point	Read Only	Remarks
/xxx	No	Directory where a dedicated resource pool mounts an SFS disk. You can specify this directory.
/home/ma-user/modelarts	No	This folder is empty. You should use it as the main directory.
/cache	No	Used to mount the hard disk of the host NVMe (supported by bare metal specifications).
/dev/shm	No	Used for PyTorch engine acceleration
/usr/local/nvidia	Yes	NV library of the host machine.

FAQs

1. What Should I Do If an Error Is Reported Indicating that the .so File in the \$ANACONDA_DIR/envs/\$DEFAULT_CONDA_ENV_NAME/lib Directory Cannot Be Found During Training?

If there is an error indicating that the .so file is unavailable in the **\$ANACONDA_DIR/envs/\$DEFAULT_CONDA_ENV_NAME/lib** directory, add the directory to **LD_LIBRARY_PATH** and place the following command before the preceding boot command:

```
export LD_LIBRARY_PATH=$ANACONDA_DIR/envs/$DEFAULT_CONDA_ENV_NAME/  
lib:$LD_LIBRARY_PATH;
```

For example, the example boot command used in method 1 is as follows:

```
export LD_LIBRARY_PATH=$ANACONDA_DIR/envs/$DEFAULT_CONDA_ENV_NAME/  
lib:$LD_LIBRARY_PATH; python /home/ma-user/modelarts/user-job-dir/code/train.py
```

7.2.4 Configuring Password-free SSH Mutual Trust Between Instances for a Training Job Created Using a Custom Image

For distributed training with custom images using MPI or Horovod, set up password-free SSH trust between instances to enable seamless communication. Otherwise, the training will fail.

This involves code adaptation and training job parameter configuration.

1. Create a custom image with OpenSSH pre-installed. The training framework should be MPI or Horovod.
2. Create a boot script file **start_sshd.sh**.

```
MY_SSHD_PORT=${MY_SSHD_PORT:-"38888"}  
mkdir -p /home/ma-user/etc  
ssh-keygen -f /home/ma-user/etc/ssh_host_rsa_key0 -N "" -t rsa > /dev/null  
/usr/sbin/sshd -p $MY_SSHD_PORT -h /home/ma-user/etc/ssh_host_rsa_key0
```
3. Upload the sshd startup script file to the training code directory in OBS.
4. Create a training job using the custom image.

- **Code Directory:** Select the OBS path where the sshd boot script file is stored.
 - **Boot Command:** Adapt the boot command to the sshd boot script.
`bash ${MA_JOB_DIR}/demo-code/start_sshd.sh && your custom command`
In the command, **your custom command** indicates custom commands you want to execute in the training job.
 - **Environment Variable:** Add **MY_SSHD_PORT = 38888**.
 - **Password-free SSH Between Nodes:** Enable it and set **Password-free SSH File Directory**. Use the default value in most cases. After a training job is delivered, the SSH key file and configuration file **authorized_keys config id_rsa id_rsa.pub** are automatically generated in the **/home/ma-user/.ssh** directory of the training container.
5. After a training job is created, its instances can establish an SSH connection with each other by using the domain name and port number throughout the training process. The sample code is as follows:
- ```
ssh modelarts-job-a0978141-1712-4f9b-8a83-000000000000-worker-1 -p $MY_SSHD_PORT
```

## 7.3 Preparing a Model Training Image

ModelArts provides deep learning-powered base images such as TensorFlow, PyTorch, and MindSpore images. In these images, the software mandatory for running training jobs has been installed. If the software in the base images cannot meet your service requirements, create new images based on the base images and use the new images to create training jobs.

### Preset Training Images

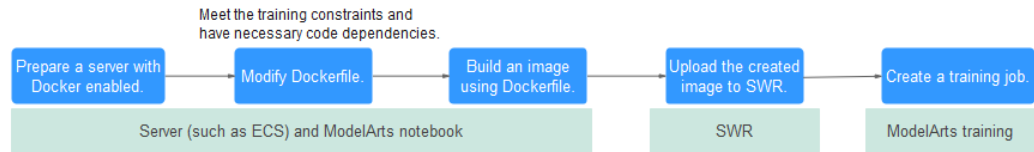
The following table lists the preset training base images of ModelArts.

**Table 7-9** ModelArts training base images

| Engine                | Version                                                              |
|-----------------------|----------------------------------------------------------------------|
| PyTorch               | pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64                   |
| TensorFlow            | tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64                |
| Horovod               | horovod_0.20.0-tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64 |
|                       | horovod_0.22.1-pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64    |
| MPI                   | mindspore_1.3.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64                 |
| Ascend-Powered-Engine | tensorflow_1.15-cann_5.1.0-py_3.7-euler_2.8.3-aarch64                |
|                       | mindspore_1.7.0-cann_5.1.0-py_3.7-euler_2.8.3-aarch64                |

## Procedure

**Figure 7-4** Creating a custom image for a training job



**Scenario 1:** If the preset images meet ModelArts training constraints but lack necessary code dependencies, install additional software packages.

For details, see [Creating a Custom Training Image Using a Preset Image](#).

**Scenario 2:** If the local images meet code dependency requirements but not ModelArts training constraints, adapt them to ModelArts.

For details, see [Migrating Existing Images to ModelArts](#).

**Scenario 3:** If neither the preset nor local images meet your needs, create an image that has necessary code dependencies and meet ModelArts constraints. For details, see the following cases:

[Creating a Custom Training Image \(PyTorch + CPU/GPU\)](#)

[Creating a Custom Training Image \(MPI + CPU/GPU\)](#)

[Creating a Custom Training Image \(Tensorflow + GPU\)](#)

## Installing pip Dependencies in an Image

Before creating distributed training jobs, pre-install all required pip dependencies. If there are more than 10 nodes, the system automatically deletes the pip source configuration. Executing pip install commands during training may cause training failures.

Install all dependency packages before training. This stops failures from missing pip source configurations when using many nodes, making training more stable and efficient.

Install pip dependencies in either of the following ways:

- Method 1: Install dependencies in a notebook and save it as an image.
  - a. Run the target image in the notebook environment.
  - b. Run the **pip install** command to install all dependency packages.
  - c. Save the image.
  - d. Obtain the SWR address on the image details page for future training.
- Method 2: Install dependencies in a local container and export an image.
  - a. Run the container in the local environment and run the **pip install** command to install all dependencies.
  - b. Run the following command to save the container as an image:

```
docker commit <Container ID> <Image name>
```

Example:



```
docker commit my_container my_image:v1
```

- c. Run the following command to export the image as a .tar file:

```
docker save -o <tar file name>.tar <Image name>:<Tag>
```

Example:

```
docker save -o my_image_v1.tar my_image:v1
```

- d. Upload the image to SWR for future training jobs.

## 7.4 Creating an Algorithm

Machine learning explores general rules from limited volume of data and uses these rules to predict unknown data. To obtain more accurate prediction results, select a proper algorithm to train your model. ModelArts provides a large number of algorithm samples for different scenarios. This section describes algorithm sources and learning modes.

### Algorithm Sources

ModelArts provides the following algorithm sources for model training:

- Using a subscribed algorithm  
You can directly subscribe to algorithms in ModelArts AI Gallery and use them to build models without writing code.
- Using a preset image  
To use a custom algorithm, use a framework built in ModelArts. ModelArts supports most mainstream AI engines. For details, see [Starting Training Using a Preset Image's Boot File](#). These built-in engines pre-load some extra Python packages, such as NumPy. You can also use the **requirements.txt** file in the code directory to install dependency packages. For details about how to create a training job using a preset image, see [Developing Code for Training Using a Preset Image](#).
- Using a preset image with customization  
If you use a preset image to create an algorithm and you need to modify or add some software dependencies based on the preset image, you can customize the preset image. In this case, select a preset image and choose **Customize** from the framework version drop-down list box.  
The only difference between this method and creating an algorithm totally based on a preset image is that you must select an image. You can create a custom image based on a preset image.
- Using a custom image  
The subscribed algorithms and preset images can be used in most training scenarios. In certain scenarios, ModelArts allows you to create custom images to train models. You can create an image based on the ModelArts image specifications, select your own image and configure the code directory (optional) and boot command to create a training job.  
Custom images can be used to train models in ModelArts only after they are uploaded to Software Repository for Container (SWR). For details, see [Creating a Custom Training Image](#). Customizing an image requires a deep understanding of containers. Use this method only if the subscribed algorithms and custom scripts cannot meet your requirements.

**NOTE**

When you use a custom image to create a training job, the boot command must be executed in the `/home/ma-user` directory. Otherwise, the training job may run abnormally.

## Creating an Algorithm

Your locally developed algorithms or algorithms developed using other tools can be uploaded to ModelArts for unified management.

1. Make preparations.
  - Create a dataset in ModelArts or upload a training dataset to an OBS directory.
  - Your training script has been uploaded to an OBS directory. For details about how to develop a training script, see [Developing Code for Training Using a Preset Image](#) or [Developing Code for Training Using a Custom Image](#).
  - Create at least one empty folder in OBS for storing training outputs.
  - Make sure your OBS directory and ModelArts are in the same region.
2. Access the algorithm creation page.
  - a. Log in to the [ModelArts console](#). In the navigation pane, choose **Asset Management > Algorithm Management**.
  - b. In the **My algorithm** tab, click **Create Algorithm**. Enter the basic algorithm information, including **Name** and **Description**.
3. Set the algorithm boot mode. The options are as follows:
  - Using a preset image

**Figure 7-5** Using a preset image to create an algorithm

The screenshot shows the 'Create Algorithm' form in the ModelArts console. At the top, there are two tabs: 'Preset image' (selected) and 'Custom image'. Below the tabs, there are two dropdown menus for selecting a preset image. The first dropdown is labeled 'Preset image' and the second is labeled 'Custom image'. Below these, there are two input fields: 'Code Directory' and 'Boot File'. Each input field has a 'Select' button to its right. The 'Code Directory' field has a red asterisk and a question mark icon. The 'Boot File' field also has a red asterisk and a question mark icon.

Set **Code Directory** and **Boot File** based on the algorithm code. Ensure that the preset image you select is the same as the one you use for editing algorithm code. For example, if TensorFlow is used for writing algorithm code, select TensorFlow when you create an algorithm.

**Table 7-10** Parameters

| Parameter      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Boot Mode      | Select <b>Preset image</b> .<br>Select a preset image and its version used by the algorithm.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Code Directory | Select an OBS path for storing the algorithm code. The files required for training, such as the training code, dependency installation packages, and pre-generated models, are uploaded to the code directory.<br>Do not store training data in the code directory. When the training job starts, the data stored in the code directory will be downloaded to the backend. A large amount of training data may lead to a download failure.<br>After you create the training job, ModelArts downloads the code directory and its subdirectories to the training container.<br>Take OBS path <b>obs://obs-bucket/training-test/demo-code</b> as an example. The content in the OBS path will be automatically downloaded to <b>\${MA_JOB_DIR}/demo-code</b> in the training container, and <b>demo-code</b> (customizable) is the last-level directory of the OBS path.<br><b>NOTE</b> <ul style="list-style-type: none"><li>Any programming language is supported.</li><li>The total number of both files and folders cannot exceed 1,000.</li><li>The total size of files cannot exceed 5 GB.</li></ul> |
| Boot File      | The file must be stored in the code directory and end with .py. ModelArts supports boot files edited only in Python.<br>The boot file in the code directory is used to start a training job.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |

- Using a preset image with customization

**Figure 7-6** Creating an algorithm using a preset image with customization

★ Boot Mode

1 **Preset image** Custom image

2 3 Customize

★ Image  **Select**

★ Code Directory  **Select**

★ Boot File  **Select**

Set **Image**, **Code Directory**, and **Boot File** based on the algorithm code. Ensure that the preset image you select is the same as the one you use for editing algorithm code. For example, if TensorFlow is used for writing algorithm code, select TensorFlow when you create an algorithm.

**Table 7-11** Parameters

| Parameter | Description                                                                                                                         |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------|
| Boot Mode | Select <b>Preset image</b> .<br>Select <b>Customize</b> for the engine version.                                                     |
| Image     | Select your image uploaded to SWR. For details about how to create an image, see <a href="#">Creating a Custom Training Image</a> . |

| Parameter      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Code Directory | <p>Select an OBS path for storing the algorithm code. The files required for training, such as the training code, dependency installation packages, and pre-generated models, are uploaded to the code directory.</p> <p>Do not store training data in the code directory. When the training job starts, the data stored in the code directory will be downloaded to the backend. A large amount of training data may lead to a download failure.</p> <p>When the training job starts, ModelArts downloads the training code directory and its subdirectories to the training container.</p> <p>Take OBS path <b>obs://obs-bucket/training-test/demo-code</b> as an example. The content in the OBS path will be automatically downloaded to <b>\${MA_JOB_DIR}/demo-code</b> in the training container, and <b>demo-code</b> (customizable) is the last-level directory of the OBS path.</p> <p><b>NOTE</b></p> <ul style="list-style-type: none"><li>Any programming language is supported for training code. The training boot file must be a Python file.</li><li>The total number of both files and folders cannot exceed 1,000.</li><li>The total size of files cannot exceed 5 GB.</li><li>The file depth cannot exceed 32.</li></ul> |
| Boot File      | <p>The file must be stored in the code directory and end with .py. ModelArts supports boot files edited only in Python.</p> <p>The boot file in the code directory is used to start a training job.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

Selecting a preset image with customization results in the same background behavior as running a training job directly with that image. For example:

- The system automatically injects environment variables.  

```
PATH=${MA_HOME}/anaconda/bin:${PATH}
LD_LIBRARY_PATH=${MA_HOME}/anaconda/lib:${LD_LIBRARY_PATH}
PYTHONPATH=${MA_JOB_DIR}:${PYTHONPATH}
```
- The selected boot file will be automatically started using Python commands. Ensure that the Python environment is correct. The **PATH** environment variable is automatically injected. Run the following commands to check the Python version for the training job:  

```
export MA_HOME=/home/ma-user; docker run --rm {image} ${MA_HOME}/anaconda/bin/python -V
docker run --rm {image} $(which python) -V
```

- The system automatically adds hyperparameters associated with the preset image.
- Using a custom image

**Figure 7-7** Creating an algorithm using a custom image

The screenshot shows the 'Boot Mode' section of the ModelArts console. There are two buttons: 'Preset image' and 'Custom image'. The 'Custom image' button is highlighted with a red rectangular box. Below these buttons are three input fields: 'Image', 'Code Directory', and 'Boot Command'. Each field has a 'Select' button to its right. The 'Boot Command' field also has a question mark icon next to its label.

**Table 7-12** Parameters

| Parameter | Description                                                                                                                         |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------|
| Boot Mode | Select <b>Custom image</b> .                                                                                                        |
| Image     | Select your image uploaded to SWR. For details about how to create an image, see <a href="#">Creating a Custom Training Image</a> . |

| Parameter      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Code Directory | <p>Select an OBS path for storing the algorithm code. The files required for training, such as the training code, dependency installation packages, and pre-generated models, are uploaded to the code directory. Configure this parameter only if your custom image does not contain training code.</p> <p>Do not store training data in the code directory. When the training job starts, the data stored in the code directory will be downloaded to the backend. A large amount of training data may lead to a download failure.</p> <p>When the training job starts, ModelArts downloads the training code directory and its subdirectories to the training container.</p> <p>Take OBS path <b>obs://obs-bucket/training-test/demo-code</b> as an example. The content in the OBS path will be automatically downloaded to <b>\${MA_JOB_DIR}/demo-code</b> in the training container, and <b>demo-code</b> (customizable) is the last-level directory of the OBS path.</p> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>Any programming language is supported for training code. The training boot file must be a Python file.</li> <li>The total number of both files and folders cannot exceed 1,000.</li> <li>The total size of files cannot exceed 5 GB.</li> <li>The file depth cannot exceed 32.</li> </ul> |

| Parameter    | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Boot Command | <p>Command for booting an image. This parameter is mandatory.</p> <p>When a training job is running, the boot command is automatically executed after the code directory is downloaded.</p> <ul style="list-style-type: none"><li>• If the training boot script is a .py file, <b>train.py</b> for example, the boot command is as follows:<br/><pre>python \${MA_JOB_DIR}/demo-code/train.py</pre></li><li>• If the training boot script is a .sh file, <b>main.sh</b> for example, the boot command is as follows:<br/><pre>bash \${MA_JOB_DIR}/demo-code/main.sh</pre></li></ul> <p>You can use semicolons (;) and ampersands (&amp;&amp;) to combine multiple commands. <b>demo-code</b> in the command is the last-level OBS directory where the code is stored. Replace it with the actual one.</p> <p>If there are input pipes, output pipes, or hyperparameters, ensure that the last command of the boot command runs the training script.</p> <p>Reason: The system appends input pipes, output pipes, and hyperparameters to the end of the boot command. If the last command is not the training script, an error will occur.</p> <p>Example: If the last line of the boot command is <b>python train.py</b> and the <b>--data_url</b> hyperparameter exists, the system executes <b>python train.py --data_url=/input</b> when running properly. However, if the boot command ends with another command, such as:</p> <pre>python train.py<br/>pwd # The last command is pwd instead of the training script.</pre> <p>The system will execute <b>python train.py pwd --data_url=/input</b>, leading to an error.</p> |

For details about how to use custom images supported by training, see [Boot Command Specifications for Custom Images](#).

4. Configure pipelines.

An algorithm obtains data from an OBS bucket or dataset for model training. The training output is stored in an OBS bucket. The input and output parameters in your algorithm code must be parsed to enable data exchange between ModelArts and OBS. For details about how to develop code for training on ModelArts, see [Preparing Model Training Code](#).

- Input configurations



**Table 7-13** Input configurations

| Parameter      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Parameter Name | <p>Set this parameter based on the data input parameter in your algorithm code. The code path parameter must be the same as the training input parameter parsed in your algorithm code. Otherwise, the algorithm code cannot obtain the input data.</p> <p>For example, If you use <b>argparse</b> in the algorithm code to parse <b>data_url</b> into the data input, set the data input parameter to <b>data_url</b> when creating the algorithm.</p>                                                                                                                                                                                                                                                                                                                                                            |
| Description    | Customize the description of the input parameter.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Obtained from  | Select a source of the input parameter, <b>Hyperparameters</b> (default) or <b>Environment variables</b> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Constraints    | <p>Enable this parameter to specify the input source. You can select a storage path or ModelArts dataset. This parameter is optional.</p> <p>If you select a ModelArts dataset, set the following parameters:</p> <ul style="list-style-type: none"><li>• <b>Labeling Type</b>: For details, see <a href="#">Creating a Manual Labeling Job</a>.</li><li>• <b>Data Format</b>, which can be <b>Default</b>, <b>CarbonData</b>, or both. <b>Default</b> indicates the manifest format.</li><li>• <b>Data Segmentation</b> is available only for image classification, object detection, text classification, and sound classification datasets. The options are <b>Segmented dataset</b>, <b>Dataset not segmented</b>, and <b>Unlimited</b>. For details, see <a href="#">Publishing a Data Version</a>.</li></ul> |
| Add            | Add multiple input data sources based on your algorithm.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

– Output configurations

**Table 7-14** Output configurations

| Parameter      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Parameter Name | Set this parameter based on the data output parameter in your algorithm code. The code path parameter must be the same as the data output parameter parsed in your algorithm code. Otherwise, the algorithm code cannot obtain the output path.<br><br>For example, if you use <b>argparse</b> in the algorithm code to parse <b>train_url</b> into the data output, set the data output parameter to <b>train_url</b> when creating the algorithm. |
| Description    | Customize the description of the output parameter.                                                                                                                                                                                                                                                                                                                                                                                                  |
| Obtained from  | Select a source of the output parameter, <b>Hyperparameters</b> (default) or <b>Environment variables</b> .                                                                                                                                                                                                                                                                                                                                         |
| Add            | Add multiple output data paths based on your algorithm.                                                                                                                                                                                                                                                                                                                                                                                             |

## 5. Define hyperparameters.

When you create an algorithm, ModelArts allows you to customize hyperparameters so you can view or modify them anytime. Defined hyperparameters are displayed in the boot command and passed to your boot file as CLI parameters.

- a. Click **Add hyperparameter** to manually add hyperparameters.
- b. Edit hyperparameters.

 **NOTE**

To ensure data security, do not enter sensitive information, such as plaintext passwords.

**Table 7-15** Hyperparameter parameters

| Parameter  | Description                                                                                                                     |
|------------|---------------------------------------------------------------------------------------------------------------------------------|
| Name       | Enter the hyperparameter name.<br>Enter 1 to 64 characters. Only letters, digits, hyphens (-), and underscores (_) are allowed. |
| Type       | Select the data type of the hyperparameter. The value can be <b>String</b> , <b>Integer</b> , <b>Float</b> , or <b>Boolean</b>  |
| Default    | Set the default value of the hyperparameter. This value will be used for training jobs by default.                              |
| Restrained | Click <b>Restrained</b> . Then, set the range of the default value or enumerated value in the dialog box displayed.             |

| Parameter   | Description                                                                                                                                                                                                                                                                                                                                                                                    |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Required    | Select <b>Yes</b> or <b>No</b> . <ul style="list-style-type: none"><li>If you select <b>No</b>, you can delete the hyperparameter on the training job creation page when using this algorithm to create a training job.</li><li>If you select <b>Yes</b>, you cannot delete the hyperparameter on the training job creation page when using this algorithm to create a training job.</li></ul> |
| Description | Enter the description of the hyperparameter.<br>Only letters, digits, spaces, hyphens (-), underscores (_), commas (,), and periods (.) are allowed.                                                                                                                                                                                                                                           |

6. Configure supported policies.

Auto search on ModelArts automatically finds the optimal hyperparameters without any code modification. For details about parameter settings, see [Creating a Training Job for Automatic Model Tuning](#).

Only the **tensorflow\_2.1.0-cuda\_10.1-py\_3.7-ubuntu\_18.04-x86\_64** and **pytorch\_1.8.0-cuda\_10.2-py\_3.7-ubuntu\_18.04-x86\_64** images are available for auto search.

7. Add training constraints.

You can add training constraints of the algorithm based on your needs.

- **Resource Type:** Select the required resource types.
- **Multicard Training:** Choose whether to support multi-PU training.
- **Distributed Training:** Choose whether to support distributed training.

8. Click **Submit**.

In the algorithm list, click the algorithm to access its details page and view the algorithm details.

- In the **Basic Information** tab, you can view the algorithm information.  
In the **Basic Information** tab, click **Edit** to modify algorithm information except the name and ID. After the modification, click **Save**.
- In the **Training** tab, you can view the information about the training jobs that use the algorithm, such as the training job name and status.

## Previewing the Runtime Environment



When creating an algorithm, click the arrow on  in the lower right corner of the page to know the paths of the code directory, boot file, and input and output data in the training container.

## Deleting an Algorithm

### NOTICE

Deleted algorithm assets cannot be restored.

To delete your algorithm, choose **Asset Management > Algorithm Management**. Click **Delete** in the **Operation** column. In the displayed dialog box, enter **DELETE**, and click **OK** to confirm the deletion.

To delete a subscribed algorithm, go to AI Gallery, choose **My Assets > Algorithm**, click **My Subscription**, and click **Cancel Subscription** for the algorithm you want to delete. In the displayed dialog box, click **OK**.

## 7.5 Creating a Production Training Job

Developing models involves optimizing their performance effectively. Traditional methods require repeatedly testing various model designs, datasets, and hyperparameters, which takes significant time and effort but may still fail to deliver good results. ModelArts simplifies this process by offering tools for creating training jobs, tracking progress in real time, and managing versions. With ModelArts, users can test different configurations easily and identify the best-performing setup faster.

Create a production training job in either of the following ways:

- Use the ModelArts Standard console. For details, see the following sections.
- Use the ModelArts API to create a production training job. For details, see [Using PyTorch to Create a Training Job \(New-Version Training\)](#).

### Notes and Constraints

By default, up to 10,000 training jobs can be created. You can view the remaining quota on the training job list page.

**Figure 7-8** Viewing the remaining quota of a training job

#### Training Jobs

A maximum of 10000 training jobs can be created. You can create 973 more.

### Prerequisites

- Data for training uploaded to an OBS directory.
- At least one empty folder in OBS for storing training output.

#### NOTE

ModelArts does not support encrypted OBS buckets. When creating an OBS bucket, do not enable bucket encryption.

- Account not in arrears (paid resources required for training jobs).
- OBS directory and ModelArts in the same region.
- Access authorization configured. If it is not configured, configure it by referring to [Configuring Agency Authorization for ModelArts with One Click](#).
- Training algorithm. For details, see [Creating an Algorithm](#).

## Billing

Model training in ModelArts uses compute and storage resources, which are billed. Compute resources are billed for running training jobs. Storage resources are billed for storing data in OBS or SFS. For details, see [Model Training Billing Items](#).

## Procedure

To create a training job, follow these steps:

- Step 1** Follow the steps in [Accessing the Page for Creating a Training Job](#).
- Step 2** Follow the steps in [Configuring Basic Information](#).
- Step 3** Select an algorithm type.
- Use an existing algorithm to create a training job by referring to [Choosing an Algorithm Type \(My Algorithm\)](#).
  - Use a preset image to create a training job by referring to [Choosing an Algorithm Type \(Custom Algorithm\)](#).
  - Use a custom image to create a training job by referring to [Choosing a Boot Mode \(Custom Image\)](#).
- Step 4** Configure training parameters, including the input, output, hyperparameters, and environment variables. For details, see [Configuring Training Parameters](#).
- Step 5** Select a resource pool as needed. A dedicated resource pool is recommended for optimal performance. For details about the differences between dedicated and public resource pools, see [Differences Between Dedicated Resource Pools and Public Resource Pools](#).
- [Configuring a Public Resource Pool](#)
  - [Configuring a Dedicated Resource Pool](#)
- Step 6** Select a training mode when you use a preset MindSpore engine and Ascend resources. For details, see [\(Optional\) Selecting a Training Mode](#).
- Step 7** Add tags if you want to manage training jobs by group. For details, see [\(Optional\) Adding Tags](#).
- Step 8** Perform follow-up procedure. For details, see [Follow-Up Operations](#).

----End

## Accessing the Page for Creating a Training Job

1. Log in to the [ModelArts console](#).
2. In the navigation pane, choose **Model Training** > **Training Jobs**.
3. Click **Create Training Job** and click **Old Version** in the upper right corner. The old version of the **Create Training Job** page is displayed.

## Configuring Basic Information

On the **Create Training Job** page, configure parameters.

**Table 7-16** Basic information

| Parameter   | Description                                                                                                                                                                                                                                                                                                                                         |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Name        | Job name, which is mandatory.<br>The system automatically generates a name, which you can then rename according to the following rules. <ul style="list-style-type: none"><li>• The name contains 1 to 64 characters.</li><li>• Letters, digits, hyphens (-), and underscores (_) are allowed.</li></ul>                                            |
| Description | Job description, which helps you learn about the job information in the training job list.                                                                                                                                                                                                                                                          |
| Experiment  | Experiment for classifying and managing the job. <ul style="list-style-type: none"><li>• If you select <b>Create new</b>, enter the experiment name and description.</li><li>• If you select <b>Use existing</b>, select an experiment name.</li><li>• If you select <b>Not required</b>, this job will not be managed in any experiment.</li></ul> |

## Choosing an Algorithm Type (My Algorithm)

Set **Algorithm Type** to **My algorithm** and select an algorithm from the algorithm list. If no algorithm meets the requirements, you can create an algorithm. For details, see [Creating an Algorithm](#).

## Choosing an Algorithm Type (Custom Algorithm)

If an algorithm is available in algorithm management, choose **My algorithm**. If no algorithm is available, choose **Custom algorithm**. If you use a custom algorithm to create a training job, select a boot mode by referring to [Table 7-17](#).

**Table 7-17** Creating a training job using a custom algorithm

| Parameter      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Algorithm Type | Select <b>Custom algorithm</b> . This parameter is mandatory.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Boot Mode      | Select <b>Preset image</b> and select the preset image engine and engine version to be used by the training job.<br><br>If you select <b>Customize</b> for the engine version, select a custom image from <b>Image</b> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Image          | This parameter is displayed and mandatory only when the preset image version is set to <b>Customize</b> .<br>You can set the container image path in either of the following ways: <ul style="list-style-type: none"> <li>To select your image or an image shared by others, click <b>Select</b> on the right and select a container image for training. The required image must be uploaded to SWR beforehand.</li> <li>To select a public image, enter the address of the public image in SWR. Enter the image path in the format of "Organization name/Image name:Version name". Do not contain the domain name in the path because the system will automatically add the domain name to the path.</li> </ul>                                                                                                                                        |
| Code Source    | Select a training code source. <ul style="list-style-type: none"> <li>OBS: Select <b>OBS</b> if the training code is stored in an OBS bucket.</li> <li>SFS: Select <b>SFS</b> if the training code is stored in an SFS file system.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| Code Directory | This parameter is available only when <b>Code Source</b> is set to <b>OBS</b> .<br>Select the OBS directory where the training code file is stored. This parameter is mandatory. <ul style="list-style-type: none"> <li>Upload code to the OBS bucket beforehand. The total size of files in the directory cannot exceed 5 GB, the number of files cannot exceed 1000, and the folder depth cannot exceed 32.</li> <li>The training code file is automatically downloaded to the <b>\${MA_JOB_DIR}/demo-code</b> directory of the training container when the training job is started. <b>demo-code</b> is the last-level OBS directory for storing the code. For example, if <b>Code Directory</b> is set to <b>/test/code</b>, the training code file is downloaded to the <b>\${MA_JOB_DIR}/code</b> directory of the training container.</li> </ul> |

| Parameter            | Description                                                                                                                                                                                                                                                                                                                                               |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Boot File            | Select the Python boot script of the training job in the code directory. This parameter is mandatory.<br>ModelArts supports only the boot file written in Python. Therefore, the boot file must end with .py.                                                                                                                                             |
| Local Code Directory | This parameter is available only when <b>Code Source</b> is set to <b>OBS</b> .<br>Specify the local directory of a training container. When a training starts, the system automatically downloads the code directory to this directory.<br>The default local code directory is <b>/home/ma-user/modelarts/user-job-dir</b> . This parameter is optional. |
| Work Directory       | During training, the system automatically runs the <b>cd</b> command to execute the boot file in this directory.                                                                                                                                                                                                                                          |

Selecting a preset image with customization results in the same background behavior as running a training job directly with that image. For example:

- The system automatically injects environment variables.  
PATH=\${MA\_HOME}/anaconda/bin:\${PATH}  
LD\_LIBRARY\_PATH=\${MA\_HOME}/anaconda/lib:\${LD\_LIBRARY\_PATH}  
PYTHONPATH=\${MA\_JOB\_DIR}:\${PYTHONPATH}
- The selected boot file will be automatically started using Python commands. Ensure that the Python environment is correct. The **PATH** environment variable is automatically injected. Run the following commands to check the Python version for the training job:  
export MA\_HOME=/home/ma-user; docker run --rm {image} \${MA\_HOME}/anaconda/bin/python -V  
docker run --rm {image} \$(which python) -V
- The system automatically adds hyperparameters associated with the preset image.

## Choosing a Boot Mode (Custom Image)

If you use a custom image to create a training job, select a boot mode by referring to [Table 7-18](#).

**Table 7-18** Creating a training job using a custom image

| Parameter      | Description                                                   |
|----------------|---------------------------------------------------------------|
| Algorithm Type | Select <b>Custom algorithm</b> . This parameter is mandatory. |
| Boot Mode      | Select <b>Custom image</b> . This parameter is mandatory.     |



| Parameter      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Image          | <p>Container image path. This parameter is mandatory. You can set the container image path in either of the following ways:</p> <ul style="list-style-type: none"> <li>To select your image or an image shared by others, click <b>Select</b> on the right and select a container image for training. The required image must be uploaded to SWR beforehand.</li> <li>To select a public image, enter the address of the public image in SWR. Enter the image path in the format of "Organization name/Image name:Version name". Do not contain the domain name in the path because the system will automatically add the domain name to the path.</li> </ul>                                                                                                                                                          |
| Code Directory | <p>OBS directory where the training code file is stored. Configure this parameter only if your custom image does not contain training code.</p> <ul style="list-style-type: none"> <li>Upload code to the OBS bucket beforehand. The total size of files in the directory cannot exceed 5 GB, the number of files cannot exceed 1000, and the folder depth cannot exceed 32.</li> <li>The training code file is automatically downloaded to the <b>\${MA_JOB_DIR}/demo-code</b> directory of the training container when the training job is started. <b>demo-code</b> is the last-level OBS directory for storing the code. For example, if <b>Code Directory</b> is set to <b>/test/code</b>, the training code file is downloaded to the <b>\${MA_JOB_DIR}/code</b> directory of the training container.</li> </ul> |
| User ID        | <p>User ID for running the container. The default value <b>1000</b> is recommended.</p> <p>If the UID needs to be specified, its value must be within the specified range. The UID ranges of different resource pools are as follows:</p> <ul style="list-style-type: none"> <li>Public resource pool: <b>1000</b> to <b>65535</b></li> <li>Dedicated resource pool: <b>0</b> to <b>65535</b></li> </ul> <p>If the user ID is set to <b>0</b>, the user in the training container is <b>root</b>.</p>                                                                                                                                                                                                                                                                                                                  |

| Parameter            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Boot Command         | <p>Command for booting an image. This parameter is mandatory.</p> <p>When a training job is running, the boot command is automatically executed after the code directory is downloaded.</p> <ul style="list-style-type: none"> <li>If the training boot script is a .py file, <b>train.py</b> for example, the boot command is as follows.<br/> <pre>python \${MA_JOB_DIR}/demo-code/train.py</pre> </li> <li>If the training boot script is a .sh file, <b>main.sh</b> for example, the boot command is as follows:<br/> <pre>bash \${MA_JOB_DIR}/demo-code/main.sh</pre> </li> </ul> <p>You can use semicolons (;) and ampersands (&amp;&amp;) to combine multiple commands. <b>demo-code</b> in the command is the last-level OBS directory where the code is stored. Replace it with the actual one.</p> <p>If there are input pipes, output pipes, or hyperparameters, ensure that the last command of the boot command runs the training script.</p> <p>Reason: The system appends input pipes, output pipes, and hyperparameters to the end of the boot command. If the last command is not the training script, an error will occur.</p> <p>Example: If the last line of the boot command is <b>python train.py</b> and the <b>--data_url</b> hyperparameter exists, the system executes <b>python train.py --data_url=/input</b> when running properly. However, if the boot command ends with another command, such as:</p> <pre>python train.py pwd # The last command is pwd instead of the training script.</pre> <p>The system will execute <b>python train.py pwd --data_url=/input</b>, leading to an error.</p> <p><b>NOTE</b><br/>To ensure data security, do not enter sensitive information, such as plaintext passwords.</p> |
| Local Code Directory | <p>This parameter is available only when <b>Code Source</b> is set to <b>OBS</b>.</p> <p>Specify the local directory of a training container. When a training starts, the system automatically downloads the code directory to this directory.</p> <p>The default local code directory is <b>/home/ma-user/modelarts/user-job-dir</b>. This parameter is optional.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Work Directory       | <p>During training, the system automatically runs the <b>cd</b> command to execute the boot file in this directory.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

For details about how to use custom images supported by training, see [Boot Command Specifications for Custom Images](#).

## Configuring Training Parameters

Data is obtained from an OBS bucket or dataset for model training. The training output can also be stored in an OBS bucket. When creating a training job, you can configure parameters such as input, output, hyperparameters, and environment variables by referring to [Table 7-19](#).

### NOTE

The input, output, and hyperparameter parameters of a training job vary depending on the algorithm type selected during training job creation. If a parameter value is dimmed, the parameter has been configured in the algorithm code and cannot be modified.

**Table 7-19** Configuring training parameters

| Parameter | Sub-Parameter  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-----------|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Input     | Parameter name | The algorithm code reads the training input data based on the input parameter name.<br>The recommended value is <b>data_url</b> . The training input parameters must match the input parameters of the selected algorithm. For details, see <a href="#">Table 7-13</a> .                                                                                                                                                                                                                                        |
|           | Dataset        | Click <b>Dataset</b> and select the target dataset and its version in the ModelArts dataset list.<br>When the training job is started, ModelArts automatically downloads the data in the input path to the training container.                                                                                                                                                                                                                                                                                  |
|           | Data path      | Click <b>Data path</b> and select the storage path to the training input data from an OBS bucket. Files must not exceed 10 GB in total size, 1,000 in number, or 1 GB per file.<br>When the training job is started, ModelArts automatically downloads the data in the input path to the training container.                                                                                                                                                                                                    |
|           | Obtained from  | The following uses training input <b>data_path</b> as an example. <ul style="list-style-type: none"><li>If you select <b>Hyperparameters</b>, use this code to obtain the data:<pre>import argparse parser = argparse.ArgumentParser() parser.add_argument('--data_path') args, unknown = parser.parse_known_args() data_path = args.data_path</pre></li><li>If you select <b>Environment variables</b>, use this code to obtain the data:<pre>import os data_path = os.getenv("data_path", "")</pre></li></ul> |

| Parameter | Sub-Parameter  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-----------|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Output    | Parameter name | <p>The algorithm code reads the training output data based on the output parameter name.</p> <p>The recommended value is <b>train_url</b>. The training output parameters must match the output parameters of the selected algorithm. For details, see <a href="#">Table 7-14</a>.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|           | Data path      | <p>Click <b>Data path</b> and select the storage path for the training output data from an OBS bucket. Files must not exceed 1 GB in total size, 128 in number, or 128 MB per file.</p> <p>During training, the system automatically synchronizes files from the local code directory of the training container to the data path.</p> <p><b>NOTE</b><br/>The data path can only be an OBS path. To prevent any issues with data storage, choose an empty directory as the data path.</p>                                                                                                                                                                                                                                                                                                                                                    |
|           | Obtained from  | <p>The following uses the training output <b>train_url</b> as an example.</p> <ul style="list-style-type: none"> <li>If you select <b>Hyperparameters</b>, use this code to obtain the data: <pre>import argparse parser = argparse.ArgumentParser() parser.add_argument('--train_url') args, unknown = parser.parse_known_args() train_url = args.train_url</pre> </li> <li>If you select <b>Environment variables</b>, use this code to obtain the data: <pre>import os train_url = os.getenv("train_url", "")</pre> </li> </ul>                                                                                                                                                                                                                                                                                                          |
|           | Predownload    | <p>Indicates whether to pre-download the files in the output directory to a local directory.</p> <ul style="list-style-type: none"> <li>If you set <b>Predownload</b> to <b>No</b>, the system does not download the files in the training output data path to a local directory of the training container when the training job is started.</li> <li>If you set <b>Predownload</b> to <b>Yes</b>, the system automatically downloads the files in the training output data path to a local directory of the training container when the training job is started. The larger the file size, the longer the download time. To avoid excessive training time, remove any unneeded files from the local code directory of the training container as soon as possible. To use <a href="#">Resumable Training</a>, select <b>Yes</b>.</li> </ul> |

| Parameter            | Sub-Parameter | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|----------------------|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Hyperparameter       | N/A           | <p>Used for training tuning. This parameter is determined by the selected algorithm. If hyperparameters have been defined in the algorithm, all hyperparameters in the algorithm are displayed.</p> <p>Hyperparameters can be modified and deleted. The status depends on the hyperparameter constraint settings in the algorithm. For details, see <a href="#">Table 7-15</a>.</p> <p>To import hyperparameters in batches, click <b>Upload</b>. You will need to fill in the hyperparameters based on the provided template. The total number of hyperparameters should not exceed 100, or the import will fail.</p> <p><b>NOTE</b><br/>To ensure data security, do not enter sensitive information, such as plaintext passwords.</p> |
| Environment Variable | N/A           | <p>Add environment variables based on service requirements. For details about the environment variables preset in the training container, see <a href="#">Managing Environment Variables of a Training Container</a>.</p> <p>To import environment variables in batches, click <b>Upload</b>. You will need to fill in the environment variables based on the provided template. The total number of environment variables should not exceed 100, or the import will fail.</p> <p><b>NOTE</b><br/>To ensure data security, do not enter sensitive information, such as plaintext passwords.</p>                                                                                                                                         |

| Parameter    | Sub-Parameter | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|--------------|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Auto Restart | N/A           | <p>Once this feature is enabled, you can set the number of restarts and whether to enable <b>Unconditional auto restart</b>.</p> <p>After you enable auto restart, ModelArts will handle any exceptions caused by environmental issues during a training job. It will either automatically handle the exception or isolate the faulty node and then restart the job, which helps to increase the success rate of the training. To avoid losing training progress and make full use of compute, ensure that your code logic supports resumable training before enabling this function. For details, see <a href="#">Resumable Training</a>.</p> <p>The value ranges from <b>1</b> to <b>128</b>. The default value is <b>3</b>. The value cannot be changed once the training job is created. Set this parameter based on your needs.</p> <p>If <b>Unconditional auto restart</b> is selected, the training job will be restarted unconditionally once the system detects a training exception. To prevent invalid restarts, it supports a maximum of three consecutive unconditional restarts.</p> <p>ModelArts continuously monitors job processes to detect suspension and optimize resource usage. When <b>Restart Upon Suspension</b> is enabled, suspended jobs can be automatically restarted at the process level. However, ModelArts does not verify code logic, and suspension detection is periodic, which may result in false reports. By enabling this feature, you acknowledge the possibility of false positives. To prevent unnecessary restarts, ModelArts limits consecutive restarts to three.</p> <p>If auto restart is triggered during training, the system records the restart information. You can check the fault recovery details on the training job details page. For details, see <a href="#">Training Job Fault Tolerance Check</a>.</p> |

## Configuring a Public Resource Pool

To configure a public resource pool, refer to [Table 7-20](#).

**Table 7-20** Configuring a public resource pool for a training job

| Parameter     | Description                          |
|---------------|--------------------------------------|
| Resource Pool | Select <b>Public resource pool</b> . |

| Parameter             | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Resource Type         | Select the resource type required for training. This parameter is mandatory. If a resource type has been defined in the training code, select a proper resource type based on algorithm constraints. For example, if the resource type defined in the training code is CPU and you select other types, the training fails. If some resource types are invisible or unavailable for selection, they are not supported.                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| Specifications        | <p>Select the required resource specifications based on the resource type.</p> <p>If <b>Data path</b> is selected for <b>Input</b>, you can click <b>Check Input Size</b> on the right to ensure the storage is larger than the input data size.</p>  <p><b>NOTICE</b><br/>The resource flavor <b>GPU:n*tnt004</b> (<i>n</i> indicates a specific number) does not support multi-process training.</p>                                                                                                                                                                                                                                                                                                                                                                              |
| Compute Nodes         | <p>Select the number of instances as required. The default value is <b>1</b>.</p> <ul style="list-style-type: none"> <li>If only one instance is used, a single-node training job is created. ModelArts starts one training container on this node. The training container exclusively uses the compute resources of the selected flavor.</li> <li>If more than one instance is used, a distributed training job is created. For more information about distributed training configurations, see <a href="#">Overview</a>. Before creating distributed training jobs, pre-install all required pip dependencies (see <a href="#">Installing pip Dependencies in an Image</a>). If there are more than 10 nodes, the system automatically deletes the pip source configuration. Executing pip install commands during training may cause training failures.</li> </ul> |
| Persistent Log Saving | <p>If you select CPU or GPU flavors, <b>Persistent Log Saving</b> is available for you to configure.</p> <ul style="list-style-type: none"> <li>After this feature is enabled (default), configure <b>Job Log Path</b>. The system permanently stores training logs to the specified OBS path.</li> <li>After this feature is disabled, ModelArts automatically stores the logs for 30 days. You can download all logs on the job details page to a local path.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                            |

| Parameter          | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Job Log Path       | <p>When enabling <b>Persistent Log Saving</b> or selecting Ascend resources, select an empty OBS directory for <b>Job Log Path</b> to store log files generated by the training job.</p> <p>Ensure that you have read and write permissions to the selected OBS directory.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Event Notification | <p>Indicates whether to enable event notification.</p> <ul style="list-style-type: none"> <li>This feature is disabled by default, which means SMN is disabled.</li> <li>After this feature is enabled, you will be notified of specific events, such as job status changes or suspected suspensions, via an SMS or email. Notifications will be billed based on SMN pricing. In this case, you must configure the topic name and events. <ul style="list-style-type: none"> <li><b>Topic:</b> topic of event notifications. Click <b>Create Topic</b> to create a topic on the SMN console.</li> <li><b>Event:</b> events you want to subscribe to. Examples: <b>JobStarted</b>, <b>JobCompleted</b>, <b>JobFailed</b>, <b>JobTerminated</b>, and <b>JobHanged</b>.</li> </ul> </li> </ul> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>After you create a topic on the SMN console, add a subscription to the topic, and confirm the subscription. Then, you will be notified of events. For details, see <a href="#">Adding a Subscription</a>.</li> <li>SMN charges you for the number of notification messages. For details, see <a href="#">Billing</a>.</li> <li>Only training jobs using GPUs or NPUs support <b>JobHanged</b> events.</li> </ul> |
| Auto Stop          | <p>When using paid resources, you can determine whether to enable auto stop.</p> <ul style="list-style-type: none"> <li>This function is disabled by default, the training job keeps running until the training is completed.</li> <li>If this function is enabled, configure the auto stop time. The value can be <b>1 hour</b>, <b>2 hours</b>, <b>4 hours</b>, <b>6 hours</b>, or <b>Customize</b>. The customized time must range from 1 hour to 720 hours. When you enable this feature, the training stops automatically when the time limit is reached. The time limit does not count down when the training is paused.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |



| Parameter                    | Description                                                                                                                                                                                                                                                                                                                                                                                         |
|------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| training_ssh_configure_nodes | <p>Whether to enable password-free SSH mutual trust between nodes.</p> <ul style="list-style-type: none"><li>• This feature is disabled by default.</li><li>• Enabling this feature requires you to configure the SSH key directory. This is where the automatically generated SSH key file will be stored within the training container. The default value is <b>/home/ma-user/.ssh</b>.</li></ul> |

## Configuring a Dedicated Resource Pool

To configure a dedicated resource pool, refer to [Table 7-21](#).

**Table 7-21** Configuring a dedicated resource pool for a training job

| Parameter      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Resource Pool  | <p>Select a dedicated resource pool.</p> <p>If you select a dedicated resource pool, you can view the name, status, node specifications, number of idle/fragmented nodes, number of available/total nodes, and number of cards of the resource pool. Hover over <b>View</b> in the <b>Idle/Fragmented Nodes</b> column to check fragment details and check whether the resource pool meets the training requirements.</p>                                                                                                                                                                         |
| Specifications | <p>Select the required resource specifications based on the resource type.</p> <p>If <b>Data path</b> is selected for <b>Input</b>, you can click <b>Check Input Size</b> on the right to ensure the storage is larger than the input data size.</p> <div><p>★ Specifications</p><div><div>32GB</div><div>Data disk size</div></div><div>Check Input Size</div><p>Ensure the storage is larger than the input data size.</p></div> <p><b>NOTE</b></p> <p>The resource specifications <b>GPU:n*<i>tnt</i>004</b> (<i>n</i> indicates a specific number) do not support multi-process training.</p> |

| Parameter     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Compute Nodes | <p>Select the number of instances as required. The default value is <b>1</b>.</p> <ul style="list-style-type: none"> <li>• If only one instance is used, a single-node training job is created. ModelArts starts one training container on this node. The training container exclusively uses the compute resources of the selected flavor.</li> <li>• If more than one instance is used, a distributed training job is created. For more information about distributed training configurations, see <a href="#">Overview</a>. Before creating distributed training jobs, pre-install all required pip dependencies (see <a href="#">Installing pip Dependencies in an Image</a>). If there are more than 10 nodes, the system automatically deletes the pip source configuration. Executing pip install commands during training may cause training failures.</li> </ul> |
| Job Priority  | <p>When using a dedicated resource pool, you can set the priority of the training job. The value ranges from 1 to 3. The default priority is <b>1</b>, and the highest priority is <b>3</b>.</p> <ul style="list-style-type: none"> <li>• By default, the job priority can be set to <b>1</b> or <b>2</b>. After the permission to <a href="#">set the highest job priority</a> is configured, the priority can be set to <b>1</b> to <b>3</b>.</li> <li>• If a training job is in the <b>Pending</b> state for a long time, you can change the job priority to reduce the queuing duration. For details, see <a href="#">Priority of a Training Job</a>.</li> </ul>                                                                                                                                                                                                      |
| Preemption    | <p>When enabled, jobs that allow preemption may be terminated and re-queued if resource pool capacity is insufficient. To avoid losing training progress, configure resumable training before enabling this function. For details, see <a href="#">Resumable Training</a>.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |

| Parameter             | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SFS Turbo             | <p>When ModelArts and SFS Turbo are directly connected, multiple SFS Turbo file systems can be mounted to a training job to store training data. Click <b>Add Mount Configuration</b> and set the following parameters:</p> <ul style="list-style-type: none"> <li>• <b>Name:</b> Select an SFS Turbo file system.</li> <li>• <b>Mount Path:</b> Enter the SFS Turbo mounting path in the training container.</li> <li>• <b>Directory:</b> Specify the SFS Turbo storage location. If you have configured the folder control permission, select a storage location. If you have not configured the folder control permission, retain the default value / or customize a location.</li> <li>• <b>Mounting Mode:</b> Permission on the mounted SFS Turbo file system. This parameter is displayed as <b>Read/Write</b> or <b>Read-only</b> based on the permission of the SFS Turbo storage location. If you have not configured the folder control permission, this parameter is unavailable.</li> <li>• <b>Mount Options:</b> Configure SFS mount parameters to accelerate and optimize training. For details about the parameters, see <a href="#">Configuring SFS Turbo Mount Options</a>. Alternatively, retain the default settings below:<br/> <pre>mountOptions: - vers=3 - timeo=600 - nolock - hard</pre> </li> </ul> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>• You can mount a file system multiple times, but each mount path must be distinct. A maximum of five disks can be mounted to a training job.</li> <li>• The mounting path cannot be a / directory or a default mounting path, such as <b>/cache</b> and <b>/home/ma-user/modelarts</b>.</li> <li>• For details about how to set permissions for SFS Turbo folders, see <a href="#">Permissions Management</a>.</li> </ul> |
| Persistent Log Saving | <p>If you select CPU or GPU flavors, <b>Persistent Log Saving</b> is available for you to configure.</p> <ul style="list-style-type: none"> <li>• After this feature is enabled (default), configure <b>Job Log Path</b>. The system permanently stores training logs to the specified OBS path.</li> <li>• After this feature is disabled, ModelArts automatically stores the logs for 30 days. You can download all logs on the job details page to a local path.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

| Parameter          | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Job Log Path       | <p>When enabling <b>Persistent Log Saving</b> or selecting Ascend resources, select an empty OBS directory for <b>Job Log Path</b> to store log files generated by the training job.</p> <p>Ensure that you have read and write permissions to the selected OBS directory.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Event Notification | <p>Indicates whether to enable event notification.</p> <ul style="list-style-type: none"> <li>This feature is disabled by default, which means SMN is disabled.</li> <li>After this feature is enabled, you will be notified of specific events, such as job status changes or suspected suspensions, via an SMS or email. Notifications will be billed based on SMN pricing. In this case, you must configure the topic name and events. <ul style="list-style-type: none"> <li><b>Topic:</b> topic of event notifications. Click <b>Create Topic</b> to create a topic on the SMN console.</li> <li><b>Event:</b> events you want to subscribe to. Examples: <b>JobStarted</b>, <b>JobCompleted</b>, <b>JobFailed</b>, <b>JobTerminated</b>, and <b>JobHanged</b>.</li> </ul> </li> </ul> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>After you create a topic on the SMN console, add a subscription to the topic, and confirm the subscription. Then, you will be notified of events. For details, see <a href="#">Adding a Subscription</a>.</li> <li>SMN charges you for the number of notification messages. For details, see <a href="#">Billing</a>.</li> <li>Only training jobs using GPUs or NPUs support <b>JobHanged</b> events.</li> </ul> |
| Auto Stop          | <p>When using paid resources, you can determine whether to enable auto stop.</p> <ul style="list-style-type: none"> <li>This feature is disabled by default, the training job keeps running until the training is completed.</li> <li>If this feature is enabled, configure the auto stop time. The value can be <b>1 hour</b>, <b>2 hours</b>, <b>4 hours</b>, <b>6 hours</b>, or <b>Customize</b>. The customized time must range from 1 hour to 720 hours. When you enable this feature, the training stops automatically when the time limit is reached. The time limit does not count down when the training is paused.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

| Parameter                    | Description                                                                                                                                                                                                                                                                                                                                                                                         |
|------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| training_ssh_configure_nodes | <p>Whether to enable password-free SSH mutual trust between nodes.</p> <ul style="list-style-type: none"><li>• This feature is disabled by default.</li><li>• Enabling this feature requires you to configure the SSH key directory. This is where the automatically generated SSH key file will be stored within the training container. The default value is <b>/home/ma-user/.ssh</b>.</li></ul> |

## (Optional) Selecting a Training Mode

Select a training mode when you use a preset MindSpore engine and Ascend resources. ModelArts provides three training modes for you to select. You can obtain different diagnosis information based on the actual scenario.

- Common mode: It is the default training scenario.
- High performance mode: In this mode, certain O&M functions will be adjusted or even disabled to accelerate the running speed, but this will deteriorate fault locating. This mode is suitable for stable networks requiring high performance.
- Fault diagnosis mode: In this mode, certain O&M functions will be enabled or adjusted to collect more information for locating faults. This mode provides fault diagnosis. You can select a diagnosis type as required.

## (Optional) Adding Tags

If you want to manage training jobs by group using tags, select **Configure Now** for **Advanced Configuration** to set tags for training jobs. For details about how to use tags, see [Using TMS Tags to Manage Resources by Group](#).

## Follow-Up Operations

After parameter setting for creating a training job, click **Submit**. On the **Confirm** dialog box, click **OK**.

A training job runs for a period of time. You can go to the training job list to view the basic information about the training job.

- In the training job list, **Status** of a newly created training job is **Pending**.
- When the status of a training job changes to **Completed**, the training job is finished, and the generated model is stored in the corresponding output path.
- If the status is **Failed** or **Abnormal**, click the job name to go to the job details page and view logs for troubleshooting.

## FAQs

1. [How Do I View the Resource Usage of a Training Job in ModelArts?](#)

## 7.6 Creating a Production Training Job (New Version)

### NOTE

ModelArts has enhanced the creation page to improve the efficiency of creating training jobs. The updated page streamlines operations and enhances the GUI display.

Developing models involves optimizing their performance effectively. Traditional methods require repeatedly testing various model designs, datasets, and hyperparameters, which takes significant time and effort but may still fail to deliver good results. ModelArts simplifies this process by offering tools for creating training jobs, tracking progress in real time, and managing versions. With ModelArts, users can test different configurations easily and identify the best-performing setup faster.

Create a production training job in either of the following ways:

- Create a production training job on the ModelArts Standard console. This chapter provides the operation guide of the new page. For details about the operation guide of the old (default) page, see [Creating a Production Training Job](#).
- Use the ModelArts API to create a production training job. For details, see [Using PyTorch to Create a Training Job \(New-Version Training\)](#).

### Notes and Constraints

By default, up to 10,000 training jobs can be created. You can view the remaining quota on the training job list page.

**Figure 7-9** Viewing the remaining quota of a training job

#### Training Jobs

A maximum of 10000 training jobs can be created. You can create 973 more.

### Prerequisites

- Account not in arrears (paid resources required for training jobs).
- Data for training uploaded to an OBS directory.
- At least one empty folder in OBS for storing training output.

### NOTE

ModelArts does not support encrypted OBS buckets. When creating an OBS bucket, do not enable bucket encryption.

- OBS directory and ModelArts in the same region.
- Access authorization configured. If it is not configured, configure it by referring to [Configuring Agency Authorization for ModelArts with One Click](#).

## Billing

Model training in ModelArts uses compute and storage resources, which are billed. Compute resources are billed for running training jobs. Storage resources are billed for storing data in OBS or SFS. For details, see [Model Training Billing Items](#).

## Procedure

To create a training job, follow these steps:

Access the page for creating a training job. For details, see [Step 1: Accessing the Page for Creating a Training Job](#).

Configure basic information. For details, see [Step 2: Configuring Basic Parameters](#).

Configure the environment. For details, see [Step 3: Configuring the Training Job Environment](#).

Configure training parameters, including inputs, outputs, hyperparameters, and environment variables. For details, see [\(Optional\) Step 4: Configuring Training Settings](#).

Configure the resource pool and instance specifications. For details, see [Step 5: Configuring Training Job Resources](#).

Configure auto restart. For details, see [Step 6: Configuring Fault Tolerance and Recovery](#).

Set the job priority, preemption, and auto stop. For details, see [Step 7: Configuring Scheduling Parameters](#).

Configure logs, event notifications, and tags. For details, see [Step 8: Configure Advanced Parameters](#).

Submit a training job and view its status. For details, see [Step 9: Submitting a Training Job and Viewing Its Status](#).

### Step 1: Accessing the Page for Creating a Training Job

1. Log in to the [ModelArts console](#).
2. In the navigation pane, choose **Model Training** > **Training Jobs**.
3. Click **Create Training Job**. The new-version page is displayed by default. The following describes how to create a training job on the old-version page.

### Step 2: Configuring Basic Parameters

On the **Create Training Job** page, configure basic parameters.

**Table 7-22** Basic parameters

| Parameter              | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Runtime Type           | Select <b>Production</b> .<br>Debug your training code either in the cloud or locally before using it to create production training jobs. <ul style="list-style-type: none"><li>• <b>Production:</b> Run the training job in the production environment. You are advised to start with the debug mode to save resources.</li><li>• <b>Debug:</b> Modify and debug your training code in real time.</li></ul>                                                                                                                                                                                                                                                                                           |
| Name                   | Job name, which is mandatory.<br>The system automatically generates a name, which you can then rename according to the following rules. <ul style="list-style-type: none"><li>• The name contains 1 to 64 characters.</li><li>• Letters, digits, hyphens (-), and underscores (_) are allowed.</li></ul>                                                                                                                                                                                                                                                                                                                                                                                               |
| Description (Optional) | Job description, which helps you learn about the job information in the training job list.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Experiment             | Specifies whether to organize training jobs into experiments for better management. It helps manage multiple job versions efficiently.<br>Experiments help manage and optimize training jobs. For example, after fine-tuning hyperparameters, you can sort and compare job results in an experiment to find the optimal training configuration. <ul style="list-style-type: none"><li>• <b>Use existing:</b> Select an existing experiment to add the training job to the experiment.</li><li>• <b>Create new:</b> Enter an experiment name and description to add the training job to the new experiment.</li></ul> If you do not enable this feature, the job will not be managed in any experiment. |

### Step 3: Configuring the Training Job Environment

When creating a training job, configure the algorithm source, boot mode, image engine and version, and code directory. Training jobs can have various algorithm types.

- **Custom algorithm:** Create a training job using a preset image or a custom image.



**Table 7-23** Environment settings (custom algorithm)

| Parameter          | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Algorithm Type     | Select <b>Custom algorithm</b> . This parameter is mandatory.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| Boot Mode          | <p>This parameter is mandatory when <b>Algorithm Type</b> is set to <b>Custom algorithm</b>. Options:</p> <ul style="list-style-type: none"> <li>• <b>Preset image</b>: Create a training job using a preset training framework and image.</li> <li>• <b>Custom image</b>: Create a training job using a custom image.</li> </ul> <p>If the software in the preset images cannot meet your needs, you can use a custom image for training. This image must be uploaded to SWR beforehand. For details about how to create an image, see <a href="#">Preparing a Model Training Image</a>.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| Engine and Version | <p>If <b>Boot Mode</b> is set to <b>Preset image</b>, you need to select the required engine and version.</p> <p>Ensure that the framework of the AI engine you select is the same as the one you use for writing algorithm code. For example, if PyTorch is used for writing algorithm code, select PyTorch when you create a job.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Image              | <p>Select a container image for training. For details about the training image creation requirements, see <a href="#">Preparing a Model Training Image</a>.</p> <ul style="list-style-type: none"> <li>• If <b>Boot Mode</b> is set to <b>Preset image</b> and the engine version is set to <b>Customize</b>, you need to select a proper image from the container images.</li> <li>• If the <b>Boot Mode</b> is set to <b>Custom image</b>, you need to select a proper image from the container images.</li> </ul> <p>You can set the container image path in either of the following ways:</p> <ul style="list-style-type: none"> <li>• To select your image or an image shared by others, click <b>Select</b> on the right and select a container image for training. The required image must be uploaded to SWR beforehand.</li> <li>• To select a public image, enter the address of the public image in SWR. Enter the image path in the format of "Organization name/Image name:Version name". Do not contain the domain name (swr.&lt;region&gt;.myhuaweicloud.com) in the path because the system will automatically add the domain name to the path. For example, if the SWR address of a public image is <b>swr.&lt;region&gt;.myhuaweicloud.com/test-image/tensorflow2_1_1:1.1.1</b>, enter <b>test-images/tensorflow2_1_1:1.1.1</b>.</li> </ul> |

| Parameter      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Code Source    | <p>Select the code source. <b>OBS</b> is selected by default.</p> <ul style="list-style-type: none"> <li>• <b>OBS</b>: Select <b>OBS</b> if the training code is stored in an OBS bucket.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Code Directory | <p>This parameter is available only when <b>Code Source</b> is set to <b>OBS</b>.</p> <p>Select the OBS directory where the training code file is stored. This parameter is mandatory when <b>Boot Mode</b> is set to <b>Preset image</b>. This parameter is optional when <b>Boot Mode</b> is set to <b>Custom image</b>.</p> <ul style="list-style-type: none"> <li>• Upload code to the OBS bucket beforehand. The total size of files in the directory cannot exceed 5 GB, the number of files cannot exceed 1,000, and the folder depth cannot exceed 32. If there is a pre-trained model, put it in the code directory.</li> <li>• The training code file is automatically downloaded to the <b>\${MA_JOB_DIR}/demo-code</b> directory of the training container when the training job is started. <b>demo-code</b> is the last-level OBS directory for storing the code. For example, if <b>Code Directory</b> is set to <b>/test/code</b>, the training code file is downloaded to the <b>\${MA_JOB_DIR}/code</b> directory of the training container.</li> </ul> |
| Boot File      | <p>Select or enter the Python boot script of the training job in the code directory. This parameter is mandatory when <b>Boot Mode</b> is set to <b>Preset image</b>. This parameter is not required when <b>Boot Mode</b> is set to <b>Custom image</b>.</p> <p>ModelArts supports only the boot file written in Python. Therefore, the boot file must end with .py.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

| Parameter    | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Boot Command | <p>Command for booting an image. This parameter is not required when <b>Boot Mode</b> is set to <b>Preset image</b>. This parameter is mandatory when <b>Boot Mode</b> is set to <b>Custom image</b>.</p> <p>When a training job is running, the boot command is automatically executed after the code directory is downloaded.</p> <ul style="list-style-type: none"> <li>If the training boot script is a .py file, <b>train.py</b> for example, the boot command is as follows.<br/> <pre>python \${MA_JOB_DIR}/demo-code/train.py</pre> </li> <li>If the training boot script is a .sh file, <b>main.sh</b> for example, the boot command is as follows:<br/> <pre>bash \${MA_JOB_DIR}/demo-code/main.sh</pre> </li> </ul> <p>You can use semicolons (;) and ampersands (&amp;&amp;) to combine multiple commands. <b>demo-code</b> in the command is the last-level OBS directory where the code is stored. Replace it with the actual one.</p> <p>If there are input pipes, output pipes, or hyperparameters, ensure that the last command of the boot command runs the training script.</p> <p>Reason: The system appends input pipes, output pipes, and hyperparameters to the end of the boot command. If the last command is not the training script, an error will occur.</p> <p>Example: If the last line of the boot command is <b>python train.py</b> and the <b>--data_url</b> hyperparameter exists, the system executes <b>python train.py --data_url=/input</b> when running properly. However, if the boot command ends with another command, such as:</p> <pre>python train.py pwd # The last command is pwd instead of the training script.</pre> <p>The system will execute <b>python train.py pwd --data_url=/input</b>, leading to an error.</p> <p><b>NOTE</b><br/>To ensure data security, do not enter sensitive information, such as plaintext passwords.</p> |
| User ID      | <p>ID of the user who runs the container. This parameter is not required when <b>Boot Mode</b> is set to <b>Preset image</b>. This parameter is optional when <b>Boot Mode</b> is set to <b>Custom image</b>.</p> <p>If the UID needs to be specified, its value must be within the specified range. The UID ranges of different resource pools are as follows:</p> <ul style="list-style-type: none"> <li>Public resource pool: <b>1000</b> to <b>65535</b></li> <li>Dedicated resource pool: <b>0</b> to <b>65535</b></li> </ul> <p>The default value <b>1000</b> is recommended.</p> <p>If the user ID is set to <b>0</b>, the user in the training container is <b>root</b>.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |

| Parameter                     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Local Code Directory          | <p>This parameter is available only when <b>Code Source</b> is set to <b>OBS</b>. This parameter is optional.</p> <p>This parameter specifies the local directory of the training container. When training starts, the code directory is downloaded to this directory. The default local code directory is <b>/home/ma-user/modelarts/user-job-dir</b>.</p> <p>Click <b>Preview Runtime Environment</b> in the upper right corner of the page to view the work directory of the training job.</p> |
| Container Execution Directory | <p>Specify the local directory of the training container. During training, the system automatically runs the <b>cd</b> command to execute the boot file in this directory.</p> <p>It is the local directory where the boot command is executed during the training job. This directory can store generated temporary files. This directory must be the parent directory of the boot file's local directory.</p>                                                                                   |

- **My algorithm:** Use an algorithm in Algorithm Management to create a training job.

Set **Algorithm Type** to **My algorithm** and select an algorithm from the algorithm list. If no algorithm meets the requirements, you can create an algorithm. For details, see [Creating an Algorithm](#).

## (Optional) Step 4: Configuring Training Settings

When creating a training job, you must configure inputs, outputs, hyperparameters, and environment variables of the training job.

**Table 7-24** Training settings

| Parameter | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Input     | <p>Click <b>Add</b> and configure training inputs.</p> <ul style="list-style-type: none"><li>• <b>Parameter name</b><br/>The algorithm code reads the training input data based on the input parameter name.<br/>The recommended value is <b>data_url</b>. The training input parameters must match the input parameters of the selected algorithm.</li><li>• <b>Input Source</b>: The training input supports OBS storage and datasets.<ul style="list-style-type: none"><li>– Click  next to <b>Input Source</b> and select the storage path to the training input data from an OBS bucket. Files must not exceed 10 GB in total size, 1,000 in number, or 1 GB per file.</li><li>– Click <b>Dataset</b> and select the target dataset and its version in the ModelArts dataset list.</li></ul></li></ul> <p>When the training job is started, ModelArts automatically downloads the data in the input path to the training container.</p> <ul style="list-style-type: none"><li>• <b>Obtained from</b><br/>The following uses training input <b>data_path</b> as an example.<ul style="list-style-type: none"><li>– If you select <b>Hyperparameters</b>, use this code to obtain the data:<pre>import argparse parser = argparse.ArgumentParser() parser.add_argument('--data_path') args, unknown = parser.parse_known_args() data_path = args.data_path</pre></li><li>– If you select <b>Environment variables</b>, use this code to obtain the data:<pre>import os data_path = os.getenv("data_path", "")</pre></li></ul></li></ul> |

| Parameter | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Output    | <p>Click <b>Add</b> and configure training outputs.</p> <ul style="list-style-type: none"> <li> <b>Parameter name</b><br/> The algorithm code reads the training output data based on the output parameter name.<br/> The recommended value is <b>train_url</b>. The training output parameters must match the output parameters of the selected algorithm. </li> <li> <b>Data path</b><br/> Click  next to <b>Data path</b> and select the storage path to the training output data from an OBS bucket. Files must not exceed 1 GB in total size, 128 in number, or 128 MB per file.<br/> During training, the system automatically synchronizes files from the local code directory of the training container to the data path.<br/> Only OBS can be used to store output data. Choose an empty folder for your output files to prevent issues with stored training inputs. </li> <li> <b>Obtained from</b><br/> The following uses the training output <b>train_url</b> as an example. <ul style="list-style-type: none"> <li> If you select <b>Hyperparameters</b>, use this code to obtain the data: <pre>import argparse parser = argparse.ArgumentParser() parser.add_argument('--train_url') args, unknown = parser.parse_known_args() train_url = args.train_url</pre> </li> <li> If you select <b>Environment variables</b>, use this code to obtain the data: <pre>import os train_url = os.getenv("train_url", "")</pre> </li> </ul> </li> <li> <b>Predownload to Container Directory</b><br/> Choose whether to pre-download files in the output directory to the training container. <ul style="list-style-type: none"> <li> If you set this parameter to <b>No</b>, the system does not download the files in the training output path to the local directory of the training container when the training job is started. </li> <li> If you set this parameter to <b>Yes</b>, the system automatically downloads the files in the training output path to the local directory of the training container when the training job is started. The larger the file size, the longer the download time. To avoid excessive training time, remove any unneeded files as soon as possible. Select <b>Yes</b> for <b>Incremental Model Training</b> or <b>Resumable Training</b>. </li> </ul> </li> </ul> |

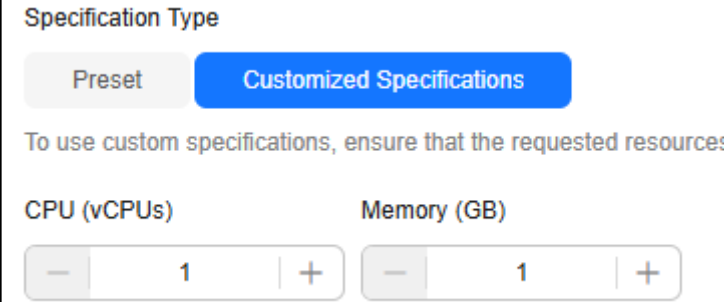
| Parameter                       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|---------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Hyperparameter                  | <p>Used for tuning. This parameter is determined by the selected algorithm. If hyperparameters have been defined in the algorithm, all hyperparameters in the algorithm are displayed.</p> <p>Hyperparameters can be modified and deleted. The status depends on the hyperparameter constraint settings in the algorithm. For details, see <a href="#">Table 7-15</a>.</p> <ul style="list-style-type: none"> <li>Click <b>Add</b> to add hyperparameters. The total number of hyperparameters cannot exceed 100.</li> <li>To import hyperparameters in batches, click <b>Upload</b>. You will need to fill in the hyperparameters based on the provided template. The total number of hyperparameters should not exceed 100, or the import will fail.</li> </ul> <p><b>NOTE</b><br/>To ensure data security, do not enter sensitive information, such as plaintext passwords.</p> |
| Environment Variable            | <p>Add environment variables based on service requirements. For details about the environment variables preset in the training container, see <a href="#">Managing Environment Variables of a Training Container</a>.</p> <ul style="list-style-type: none"> <li>Click <b>Add</b> to add environment variables. The total number of environment variables cannot exceed 100.</li> <li>To import environment variables in batches, click <b>Upload</b>. You will need to fill in the environment variables based on the provided template. The total number of environment variables should not exceed 100, or the import will fail.</li> </ul> <p><b>NOTE</b><br/>To ensure data security, do not enter sensitive information, such as plaintext passwords.</p>                                                                                                                    |
| Automated Hyperparameter Search | <p>If you select <b>My algorithm</b> for <b>Algorithm Type</b> and the selected algorithm supports the <b>autoSearch(S)</b> policy, you can click <b>More Configurations</b> to show <b>Automated Hyperparameter Search</b>.</p> <p>Selecting it enables automated hyperparameter search during training, potentially increasing the time needed. For details, see <a href="#">Overview</a>.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

## Step 5: Configuring Training Job Resources

When creating a training job, you need to choose training resources. Select **Public resource pool** or **Dedicated resource pool**. Select a resource pool as needed. A dedicated resource pool is recommended for optimal performance. For details

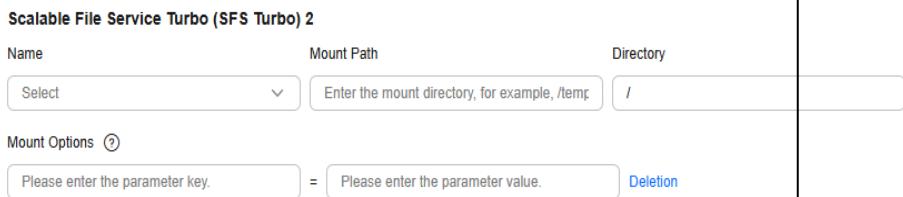
about the differences between dedicated and public resource pools, see [Differences Between Dedicated Resource Pools and Public Resource Pools](#).

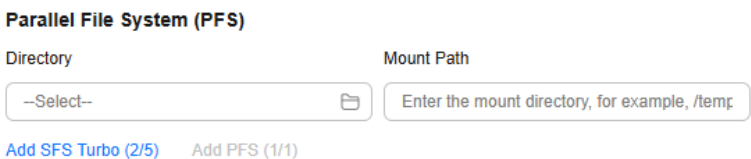
Table 7-25 Resource parameters

| Parameter           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Source of resources | <ul style="list-style-type: none"><li>• <b>Public resource pool:</b> The public resource pool is available for all tenants and does not require user creation.</li><li>• <b>Dedicated resource pool:</b> Dedicated resource pools are created separately and used exclusively. For details, see <a href="#">Creating a Standard Dedicated Resource Pool</a>.</li></ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Resource Pool       | After selecting <b>Dedicated resource pool</b> , click <b>Select Resource Pool</b> to select the target dedicated resource pool. You can view the status, node specifications, number of idle/fragmented nodes, number of available/total nodes, and number of cards of the dedicated resource pool. Hover over <b>View</b> in the <b>Idle/Fragmented Nodes</b> column to check fragment details and check whether the resource pool meets the training requirements.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| Specification Type  | <p>This parameter is displayed when you select a dedicated resource pool. The following specifications types are supported:</p> <ul style="list-style-type: none"><li>• <b>Preset:</b> Select preset instance specifications in the dedicated resource pool.</li><li>• <b>Customized Specifications:</b> Training jobs allow custom resource specifications using dedicated pools to enhance their utilization. Custom specifications must match or stay below the node specifications of the dedicated resource pool. For CPU specifications, you can only customize the number of vCPUs and memory. For GPU and Ascend specifications, you can customize the number of vCPUs, memory, and cards.</li></ul> <p><b>Figure 7-10 Specifications</b></p>  <p>The screenshot shows the 'Specifications' configuration interface. At the top, under 'Specification Type', there are two buttons: 'Preset' and 'Customized Specifications', with the latter being highlighted in blue. Below this, a text prompt states: 'To use custom specifications, ensure that the requested resource:'. Underneath, there are two input sections. The first is 'CPU (vCPUs)' with a numeric input field set to '1' and minus/plus buttons. The second is 'Memory (GB)' with a numeric input field set to '1' and minus/plus buttons.</p> |



| Parameter      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Specifications | <p>This parameter is mandatory when <b>Specification Type</b> is set to <b>Preset</b>. Select resource specifications, including server type and model.</p> <ul style="list-style-type: none"> <li>• If a resource type has been defined in the training code, select a proper resource type based on algorithm constraints. For example, if the resource type defined in the training code is CPU and you select other types, the training fails.</li> <li>• If some resource types are invisible or unavailable for selection, they are not supported.</li> <li>• If <b>Input</b> is configured, click <b>Check Input Size</b> next to resource pool specifications to ensure the storage is larger than the input data size.</li> </ul> <p><b>NOTICE</b><br/>The instance specifications <b>GPU:n*tn004</b> (<i>n</i> indicates a specific number) do not support multi-process training.</p> |
| Compute Nodes  | <p>Select the number of instances as required. The default value is <b>1</b>.</p> <ul style="list-style-type: none"> <li>• If only one instance is used, a single-node training job is created. ModelArts starts one training container on this node. The training container exclusively uses the compute resources of the selected specifications.</li> <li>• If more than one instance is used, a distributed training job is created. For more information about distributed training configurations, see <a href="#">Overview</a>.<br/>Before creating a distributed training job, pre-install all required pip dependencies (see <a href="#">Installing pip Dependencies in an Image</a>). If there are more than 10 nodes, the system automatically deletes the pip source configuration. Executing pip install commands during training may cause training failures.</li> </ul>           |

| Parameter        | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Storage Mounting | <p>When you select a dedicated resource pool, you can mount multiple storage types to improve data access efficiency.</p> <p><b>Figure 7-11 Storage mounting</b></p>  <p>• <b>Add SFS Turbo</b><br/> When ModelArts and SFS Turbo are directly connected, multiple SFS Turbo file systems can be mounted to a training job to store training data. You can mount a file system multiple times, but each mount path must be distinct. A maximum of five disks can be mounted to a training job. For details about how to configure the network connection between ModelArts and SFS Turbo, see <a href="#">Creating a Network</a>.</p> <p><b>Figure 7-12 SFS Turbo</b></p>  <ul style="list-style-type: none"> <li>– <b>Name:</b> Select an SFS Turbo file system.</li> <li>– <b>Mount Path:</b> Enter the SFS Turbo mount path in the training container. The path cannot be a / directory or a system-mounted directory like <b>/cache</b> or <b>/home/ma-user/modelarts</b>.</li> <li>– <b>Directory:</b> Specify the SFS Turbo storage location. If you have configured the folder control permission, select a storage location. If you have not configured the folder control permission, retain the default value / or customize a location.</li> <li>– <b>Mounting Mode:</b> Permission on the mounted SFS Turbo file system. This parameter is displayed as <b>Read/Write</b> or <b>Read-only</b> based on the permission of the SFS Turbo storage location. If you have not configured the folder control permission, this parameter is unavailable. For details about how to set permissions for SFS Turbo folders, see <a href="#">Permissions Management</a>.</li> <li>– <b>Mount Options:</b> Configure SFS mount parameters to accelerate and optimize training. For details about the</li> </ul> |

| Parameter                          | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                    | <p>parameters, see <a href="#">Configuring SFS Turbo Mount Options</a>. Alternatively, retain the default settings below:</p> <pre>mountOptions: - vers=3 - timeo=600 - nolock - hard</pre> <ul style="list-style-type: none"> <li>● <b>Add SFS 3.0 Capacity-Oriented:</b> A training job can mount an SFS 3.0 file system to store training data. Set the parameters below. <ul style="list-style-type: none"> <li>– <b>Name:</b> Enter the name of the file system. The name must be the same as that in SFS. Otherwise, the mounting fails.</li> <li>– <b>Mount Path:</b> Enter the cloud mount path in the training container.</li> </ul> </li> </ul> <p><b>NOTE</b><br/>To add SFS 3.0 Capacity-Oriented, submit a service ticket.</p> <ul style="list-style-type: none"> <li>● <b>Add PFS:</b> A training job can mount an OBS parallel file system to store training data. Set the parameters below.</li> </ul> <p><b>Figure 7-13 PFS</b></p>  <p><b>Parallel File System (PFS)</b></p> <p>Directory: --Select-- Mount Path: Enter the mount directory, for example, /tmp/</p> <p><a href="#">Add SFS Turbo (2/5)</a> <a href="#">Add PFS (1/1)</a></p> <ul style="list-style-type: none"> <li>– <b>Storage Configuration:</b> Select a parallel file system.</li> <li>– <b>Mount Path:</b> Enter the cloud mount path in the training container.</li> </ul> <p><b>NOTE</b><br/>To add PFS, submit a service ticket.</p> |
| Supernode Affinity Group Instances | <ul style="list-style-type: none"> <li>● Selecting a dedicated resource pool and an Snt9b23 flavor allows you to set <b>Supernode Affinity Group Instances</b>.</li> <li>● You can set this parameter if you select a supernode resource pool. If <b>Supernode Affinity Group Instances</b> is set to N, every N pods are scheduled to the same supernode to schedule affinity jobs. In distributed training, affinity job scheduling ensures uniformity in the architecture of the allocated compute resources.</li> <li>● You must set the number of instances as an integral multiple of <b>Supernode Affinity Group Instances</b>. Otherwise, the training job cannot be created.</li> <li>● For more information about supernode affinity group instances, see <a href="#">Configuring Supernode Affinity Group Instances</a>.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |

| Parameter     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Training Mode | <p>ModelArts offers various training modes when using a MindSpore preset image with Ascend specifications.</p> <ul style="list-style-type: none"><li>• Common mode: It is the default training scenario.</li><li>• High-performance mode: Certain O&amp;M functions will be adjusted or even disabled to maximally accelerate the running speed, but this will deteriorate fault locating. This mode is suitable for stable networks requiring high performance.</li><li>• Fault diagnosis mode: Certain O&amp;M functions will be enabled or adjusted to collect more information for locating faults.</li></ul> |

## Step 6: Configuring Fault Tolerance and Recovery

You can set auto restart for a training job when creating it.

**Table 7-26** Automatic restart settings

| Parameter    | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Auto Restart | <p>Choose whether to enable automatic restart for a training job.</p> <ul style="list-style-type: none"><li>• This function is disabled by default. If a training exception occurs, the job is directly stopped.</li><li>• If this function is enabled, the system will handle any exceptions caused by environmental or suspension issues during a training job. The system automatically detects faults and processes them according to the corresponding policies, thereby increasing the training success rate. Training job recovery policies enable automatic restarts at the process, container, and job levels. These policies require no manual configuration as they are automatically applied and upgraded as needed. To avoid losing training progress, ensure your code can resume training from where it is interrupted, and then enable unconditional auto restart to optimize compute usage. For details, see <a href="#">Resumable Training</a>.</li></ul> <p>If auto restart is triggered during training, the system records the restart information. You can check the fault recovery details on the training job details page. For details, see <a href="#">Training Job Fault Tolerance Check</a>.</p> |

| Parameter                  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Maximum Restarts           | <p>This parameter is available when <b>Auto Restart</b> is enabled.</p> <p>The training job will stop if it is still abnormal after maximum automatic restarts.</p> <ul style="list-style-type: none"> <li>• Default value: <b>3</b></li> <li>• Value range: <b>1 to 128</b></li> </ul> <p>The value cannot be changed once the training job is created. Set this parameter based on your needs.</p>                                                                                                                                                                    |
| Unconditional Auto Restart | <p>This parameter is available when <b>Auto Restart</b> is enabled. If <b>Unconditional auto restart</b> is selected, the training job will be restarted unconditionally once the system detects a training exception. To prevent invalid restarts, the system limits unconditional restarts to three consecutive attempts.</p>                                                                                                                                                                                                                                         |
| Restart Upon Suspension    | <p>This parameter is available when <b>Auto Restart</b> is enabled. ModelArts continuously monitors job processes to detect suspension and optimize resource usage. When this feature is enabled, suspended jobs can be automatically restarted at the process level.</p> <p>However, ModelArts does not verify code logic, and suspension detection is periodic, which may result in false reports. By enabling this feature, you acknowledge the possibility of false positives. To prevent unnecessary restarts, ModelArts limits consecutive restarts to three.</p> |

## Step 7: Configuring Scheduling Parameters

When creating a training job, you can configure its scheduling policy. For example, you can increase its priority, enable preemption, or set it to stop automatically. These changes help boost scheduling efficiency.

**Table 7-27** Scheduling settings

| Parameter                        | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Increase job scheduling priority | <ul style="list-style-type: none"> <li>Jobs join the queue based on their submission time by default. Enabling this feature increases job priorities in the queue.</li> <li>When using a dedicated resource pool, you can increase the scheduling priority of the training job. This parameter is not supported when a public resource pool is used.</li> <li>The priority can be set to <b>1</b>, <b>2</b>, or <b>3</b>. A larger number indicates a higher priority. The default priority is <b>1</b>, and the highest priority is <b>3</b>.</li> <li>To set the priority to <b>3</b>, you will also need the permission. For details about how to set the permission, see <a href="#">Assigning the Permission to Set the Highest Job Priority to an IAM User</a>.</li> <li>If a training job is in the <b>Pending</b> state for a long time, you can change the job priority to reduce the queuing duration. For details, see <a href="#">Priority of a Training Job</a>.</li> </ul> |
| Preemption                       | <p>When using a dedicated resource pool, you can set this parameter. This parameter is not supported when a public resource pool is used.</p> <p>When enabled, jobs that allow preemption may be terminated and re-queued if resource pool capacity is insufficient. To avoid losing training progress, configure resumable training before enabling this function. For details, see <a href="#">Resumable Training</a>.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Auto Stop                        | <p>Choose whether to enable <b>Auto Stop</b>.</p> <ul style="list-style-type: none"> <li>This function is disabled by default, and the training job keeps running until the training is completed.</li> <li>If you enable this function, set the auto stop time. The value can be <b>1 hour</b>, <b>2 hours</b>, <b>4 hours</b>, <b>6 hours</b>, or <b>Customize</b>. The customized time must range from 1 hour to 720 hours. When you enable this function, the training stops automatically when the time limit is reached. The time limit does not count down when the training is paused.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                |

## Step 8: Configure Advanced Parameters

**Table 7-28** Advanced settings

| Parameter             | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Persistent Log Saving | <p>This function is enabled by default when Ascend specifications are selected and cannot be modified.</p> <p>This function is available when CPU or GPU specifications are selected.</p> <ul style="list-style-type: none"> <li>If this function is enabled (default), configure <b>Log Path</b>. The platform permanently stores training logs to the specified OBS path.</li> <li>If this function is disabled, ModelArts automatically stores the logs for 30 days. You can download all logs on the job details page to a local path.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Log Path              | <p>When <b>Persistent Log Saving</b> is enabled, you must configure a log path to store log files generated by the training job.</p> <p>Ensure that you have read and write permissions to the selected OBS directory.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Event Notification    | <p>Choose whether to enable event notification for the training job.</p> <ul style="list-style-type: none"> <li>This function is deselected by default, which means SMN is disabled.</li> <li>If this function is enabled, you will be notified of specific events, such as job status changes or suspected suspensions, via an SMS or email. Notifications will be billed based on SMN pricing. In this case, you must configure the topic name and events. <ul style="list-style-type: none"> <li><b>Topic:</b> topic of event notifications. Click <b>Create Topic</b> to go to the SMN console to create a topic and add a subscription to the topic. You will receive event notifications only after the subscription status changes to <b>Confirmed</b>. For details, see <a href="#">Adding a Subscription</a>.</li> <li><b>Event:</b> events you want to subscribe to. Examples: <b>JobStarted</b>, <b>JobCompleted</b>, <b>JobFailed</b>, <b>JobTerminated</b>, <b>JobHanged</b>, <b>JobRestarted</b>, and <b>JobPreempted</b>.</li> </ul> </li> </ul> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>SMN charges you for the number of notification messages. For details, see <a href="#">Billing</a>.</li> <li>Only training jobs using GPUs or NPUs support <b>JobHanged</b> events.</li> </ul> |

| Parameter                       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|---------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Password-free SSH Between Nodes | <p>Choose whether to enable password-free SSH between nodes.</p> <ul style="list-style-type: none"><li>• This function is disabled by default, which means the password-free SSH file is not generated.</li><li>• If this function is enabled, the password-free SSH file is generated. To enable distributed training with a custom MPI or Horovod-based image, set up password-free SSH trust between nodes for seamless communication. Configure the password-free SSH file directory. This directory stores the auto-generated SSH key files in the training container. By default, it is set to <code>/home/ma-user/.ssh</code>. For details, see <a href="#">Configuring Password-free SSH Mutual Trust Between Instances for a Training Job Created Using a Custom Image</a>.</li></ul> |
| Tags                            | <p>TMS's predefined tags are recommended for adding the same tag to different cloud resources. For details about how to use tags, see <a href="#">Using TMS Tags to Manage Resources by Group</a>.</p> <p>You can add up to 20 tags to a training job.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

## Step 9: Submitting a Training Job and Viewing Its Status

After setting the parameters, click **Submit**.

A training job runs for a period of time. You can go to the training job list to view the basic information about the training job.

- In the training job list, **Status** of a newly created training job is **Pending**.
- When the status of a training job changes to **Completed**, the training job is finished, and the generated model is stored in the corresponding output path.
- If the status is **Failed** or **Abnormal**, click the job name to go to the job details page and view logs for troubleshooting.

## FAQs

1. [How Do I View the Resource Usage of a Training Job in ModelArts?](#)

## 7.7 Distributed Model Training

### 7.7.1 Overview

#### Distributed Training

**Distributed training** speeds up deep learning by running tasks simultaneously across several compute nodes like servers or GPUs, enabling faster model training



or handling bigger datasets. Distributed training splits a training task across multiple nodes, where each node processes a portion of the model. These nodes share their computed results through a communication system, enabling the full model to be trained efficiently. This approach greatly enhances training speed, particularly for complex models and large datasets.

ModelArts enables distributed training by automatically managing node communications and resources for effective parallel processing.

ModelArts provides the following capabilities:

- Extensive built-in images, meeting your requirements
- Custom development environments set up using built-in images
- Extensive tutorials, helping you quickly understand distributed training
- Distributed training debugging in development tools such as PyCharm, VS Code, and JupyterLab

It supports the following two approaches:

1. Single-node multi-PU data parallelism (DP): Multiple GPUs work together on one server to speed up training using data parallelism. This approach maximizes the use of all available GPU resources on a single server.
2. Distributed data parallelism (DDP): Multiple servers work together, with each server using several GPUs to increase training capacity. It works well for handling large datasets or complex models.

## Constraints

- If the notebook instance flavors are changed, you can only perform single-node debugging. You cannot perform distributed debugging or submit remote training jobs.
- Only the PyTorch and MindSpore AI frameworks can be used for multi-node distributed debugging. If you want to use MindSpore, each node must be equipped with eight PUs.
- The OBS paths in the debugging code should be replaced with your OBS paths.
- PyTorch is used to write debugging code in this document. The process is the same for different AI frameworks. You only need to modify some parameters.

## Billing

Model training in ModelArts uses compute and storage resources, which are billed. Compute resources are billed for running training jobs. Storage resources are billed for storing data in OBS or SFS. For details, see [Model Training Billing Items](#).

## Advantages and Disadvantages of Single-Node Multi-PU Training Using DataParallel

- Straightforward coding: Only one line of code needs to be modified.
- Bottlenecks in communication: The master GPU is used to update and distribute parameter settings, which causes high communication costs.

- Unbalanced GPU loading: The master GPU is used to summarize outputs, calculate loss, and update weights. Therefore, the GPU memory and usage are higher than those of other GPUs.

## Advantages of Multi-Node Multi-PU Training Using DistributedDataParallel

- Fast communication
- Balanced load
- Fast running speed

## Related Chapters

- **Creating a Single-Node Multi-PU Distributed Training Job (DataParallel):** describes single-node multi-PU training using DataParallel, and corresponding code modifications.
- **Creating a Multiple-Node Multi-PU Distributed Training Job (DistributedDataParallel):** describes multi-node multi-PU training using DistributedDataParallel, and corresponding code modifications.
- **Example: Creating a DDP Distributed Training Job (PyTorch + GPU):** describes the procedure and code example of distributed debugging adaptation.
- **Example: Creating a DDP Distributed Training Job (PyTorch + NPU):** provides a complete code sample of distributed parallel training for the classification task of ResNet18 on the CIFAR-10 dataset.
- **Debugging a Training Job:** describes how to use the SDK to debug a single-node or multi-node training job on the ModelArts development environment.

## 7.7.2 Creating a Single-Node Multi-PU Distributed Training Job (DataParallel)

As deep learning models grow larger, their training times increase. Efficient parallel computing methods are essential for faster training. A major challenge in single-node setups is maximizing the use of multiple GPUs. This section explains how to conduct single-node, multi-GPU data parallel training using PyTorch. Properly splitting data and syncing models across devices allows full utilization of GPU resources, significantly boosting training speed.

For details about distributed training with the MindSpore engine, visit the [MindSpore official website](#). You can select the required version in the upper left corner.

## Training Process

The process of single-node multi-PU parallel training is as follows:

1. A model is copied to multiple GPUs.
2. Data of each batch is distributed evenly to each worker GPU.
3. Each GPU does its own forward propagation and an output is obtained.
4. The master GPU with device ID 0 collects the output of each GPU and calculates the loss.

5. The master GPU distributes the loss to each worker GPU. Each GPU does its own backward propagation and calculates the gradient.
6. The master GPU collects gradients, updates parameter settings, and distributes the settings to each worker GPU.

The detailed flowchart is as follows.

**Figure 7-14** Single-node multi-PU parallel training



## Code Modifications

Model distribution: `DataParallel(model)`

The code is slightly changed and the following is a simple example:

```
import torch
class Net(torch.nn.Module):
 pass

model = Net().cuda()

DataParallel Begin
model = torch.nn.DataParallel(Net().cuda())
DataParallel End
```

### 7.7.3 Creating a Multiple-Node Multi-PU Distributed Training Job (DistributedDataParallel)

As models grow larger and datasets expand in deep learning, single-node training becomes insufficient. Implementing efficient multi-node, multi-PU training is now essential.

This section explains how to use PyTorch for multi-node, multi-PU data parallel training. It includes practical code adaptations and full examples to help you learn and apply these methods. Using ResNet18 on the CIFAR-10 dataset for image classification, this section demonstrates Distributed Data Parallel (DDP) implementation with reproducible steps for easy reference.

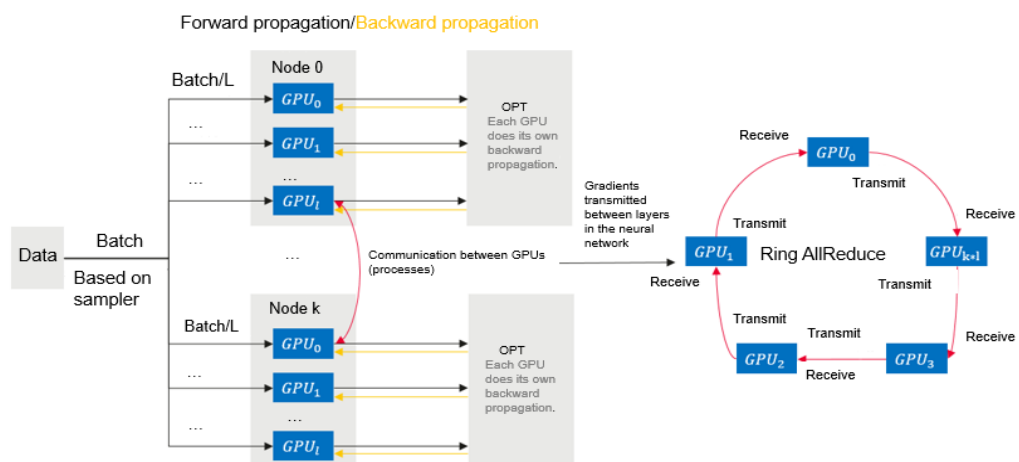
## Training Process

Compared with DataParallel, DistributedDataParallel can start multiple processes for computing, greatly improving compute resource usage. Based on **torch.distributed**, DistributedDataParallel has obvious advantages over DataParallel in the distributed computing case. The process is as follows:

1. Initializes the process group.
2. Creates a distributed parallel model. Each process has the same model and parameters.
3. Creates a distributed sampler for data distribution to enable each process to load a unique subset of the original dataset in a mini batch.
4. Parameters are organized into buckets based on their shapes or sizes, which are generally determined by each layer of the network that requires parameter update in a neural network model.
5. Each process does its own forward propagation and computes its gradient.
6. After all parameter gradients at a bucket are obtained, communication is performed for gradient averaging.
7. Each GPU updates model parameters.

The detailed flowchart is as follows.

**Figure 7-15** Multi-node multi-PU parallel training



## Code Modifications

- Multi-process startup
- New variables such as rank ID and world\_size are used along with the TCP protocol.
- Sampler for data distribution to avoid duplicate data between different processes
- Model distribution: DistributedDataParallel(model)
- Model saved in GPU 0

```
import torch
class Net(torch.nn.Module):
 pass

model = Net().cuda()

DistributedDataParallel Begin
model = torch.nn.parallel.DistributedDataParallel(Net().cuda())
DistributedDataParallel End
```

## Multi-Node Distributed Debugging Adaptation and Code Example

DDP is widely used for distributed data parallel training. Each process loads batch data independently, computes gradients, and averages them across all processes to produce the final result. Since DDP processes more samples, its gradients are more accurate. This allows for higher learning rates, speeding up model convergence.

This section offers full code examples for both single-node and distributed training using DDP for ResNet18 on the CIFAR10 dataset. The code works for multi-node distributed training and supports both CPU and GPU environments. By commenting out specific sections, you can quickly switch between distributed and single-node training modes.

The training script includes three key parameter groups: general parameters, distributed parameters, and data parameters. Distributed parameters are auto-filled by the platform, requiring no manual setup. In data parameters, a **custom\_data** toggle allows you to decide whether to train with randomly generated PyTorch data, offering flexibility for experimentation.

### CIFAR-10 dataset

In notebook instances, torchvision of the default version cannot be used to obtain datasets. Therefore, the sample code provides three training data loading methods.

Click **CIFAR-10 python version** on the [download page](#) to download the CIFAR-10 dataset.

- Download the CIFAR-10 dataset using torchvision.
- Download the CIFAR-10 dataset based on the URL and decompress the dataset in a specified directory. The sizes of the training set and test set are (50000, 3, 32, 32) and (10000, 3, 32, 32), respectively.
- Use Torch to obtain a random dataset similar to CIFAR-10. The sizes of the training set and test set are (5000, 3, 32, 32) and (1000, 3, 32, 32), respectively. The labels are still of 10 types. Set **custom\_data** to **true**, and the training task can be directly executed without loading data.

## Training code

In the following code, those commented with **### Settings for distributed training ... ###** are code modifications for multi-node distributed training.

Do not modify the sample code. After the data path is changed to your path, multi-node distributed training can be executed on ModelArts.

After the distributed code modifications are commented out, the single-node single-PU training can be executed. For details about the complete code, see [Code Example of Distributed Training](#).

- **Importing dependency packages**

```
import datetime
import inspect
import os
import pickle
import random

import argparse
import numpy as np
import torch
import torch.distributed as dist
from torch import nn, optim
from torch.utils.data import TensorDataset, DataLoader
from torch.utils.data.distributed import DistributedSampler
from sklearn.metrics import accuracy_score
```

- **Defining the method and random number for loading data** (The code for loading data is not described here due to its large amount.)

```
def setup_seed(seed):
 torch.manual_seed(seed)
 torch.cuda.manual_seed_all(seed)
 np.random.seed(seed)
 random.seed(seed)
 torch.backends.cudnn.deterministic = True

def get_data(path):
 pass
```

- **Defining a network structure**

```
class Block(nn.Module):

 def __init__(self, in_channels, out_channels, stride=1):
 super().__init__()
 self.residual_function = nn.Sequential(
 nn.Conv2d(in_channels, out_channels, kernel_size=3, stride=stride, padding=1, bias=False),
 nn.BatchNorm2d(out_channels),
 nn.ReLU(inplace=True),
 nn.Conv2d(out_channels, out_channels, kernel_size=3, padding=1, bias=False),
 nn.BatchNorm2d(out_channels)
)

 self.shortcut = nn.Sequential()
 if stride != 1 or in_channels != out_channels:
 self.shortcut = nn.Sequential(
 nn.Conv2d(in_channels, out_channels, kernel_size=1, stride=stride, bias=False),
 nn.BatchNorm2d(out_channels)
)

 def forward(self, x):
 out = self.residual_function(x) + self.shortcut(x)
 return nn.ReLU(inplace=True)(out)

class ResNet(nn.Module):

 def __init__(self, block, num_classes=10):
```

```
super().__init__()
self.conv1 = nn.Sequential(
 nn.Conv2d(3, 64, kernel_size=3, padding=1, bias=False),
 nn.BatchNorm2d(64),
 nn.ReLU(inplace=True))
self.conv2 = self.make_layer(block, 64, 64, 2, 1)
self.conv3 = self.make_layer(block, 64, 128, 2, 2)
self.conv4 = self.make_layer(block, 128, 256, 2, 2)
self.conv5 = self.make_layer(block, 256, 512, 2, 2)
self.avg_pool = nn.AdaptiveAvgPool2d((1, 1))
self.dense_layer = nn.Linear(512, num_classes)

def make_layer(self, block, in_channels, out_channels, num_blocks, stride):
 strides = [stride] + [1] * (num_blocks - 1)
 layers = []
 for stride in strides:
 layers.append(block(in_channels, out_channels, stride))
 in_channels = out_channels
 return nn.Sequential(*layers)

def forward(self, x):
 out = self.conv1(x)
 out = self.conv2(out)
 out = self.conv3(out)
 out = self.conv4(out)
 out = self.conv5(out)
 out = self.avg_pool(out)
 out = out.view(out.size(0), -1)
 out = self.dense_layer(out)
 return out
```

- **Training and validation**

```
def main():
 file_dir = os.path.dirname(inspect.getframeinfo(inspect.currentframe()).filename)

 seed = datetime.datetime.now().year
 setup_seed(seed)

 parser = argparse.ArgumentParser(description='Pytorch distribute training',
 formatter_class=argparse.ArgumentDefaultsHelpFormatter)
 parser.add_argument('--enable_gpu', default='true')
 parser.add_argument('--lr', default='0.01', help='learning rate')
 parser.add_argument('--epochs', default='100', help='training iteration')

 parser.add_argument('--init_method', default=None, help='tcp_port')
 parser.add_argument('--rank', type=int, default=0, help='index of current task')
 parser.add_argument('--world_size', type=int, default=1, help='total number of tasks')

 parser.add_argument('--custom_data', default='false')
 parser.add_argument('--data_url', type=str, default=os.path.join(file_dir, 'input_dir'))
 parser.add_argument('--output_dir', type=str, default=os.path.join(file_dir, 'output_dir'))
 args, unknown = parser.parse_known_args()

 args.enable_gpu = args.enable_gpu == 'true'
 args.custom_data = args.custom_data == 'true'
 args.lr = float(args.lr)
 args.epochs = int(args.epochs)

 if args.custom_data:
 print('[warning] you are training on custom random dataset, '
 'validation accuracy may range from 0.4 to 0.6.')

 ### Settings for distributed training. Initialize DistributedDataParallel process. The init_method,
rank, and world_size parameters are automatically input by the platform. ###
 dist.init_process_group(init_method=args.init_method, backend="nccl", world_size=args.world_size,
 rank=args.rank)
 ### Settings for distributed training. Initialize DistributedDataParallel process. The init_method,
rank, and world_size parameters are automatically input by the platform. ###

 tr_set, val_set = get_data(args.data_url, custom_data=args.custom_data)
```

```
batch_per_gpu = 128
gpus_per_node = torch.cuda.device_count() if args.enable_gpu else 1
batch = batch_per_gpu * gpus_per_node

tr_loader = DataLoader(tr_set, batch_size=batch, shuffle=False)

Settings for distributed training. Create a sampler for data distribution to ensure that different
processes load different data. ###
tr_sampler = DistributedSampler(tr_set, num_replicas=args.world_size, rank=args.rank)
tr_loader = DataLoader(tr_set, batch_size=batch, sampler=tr_sampler, shuffle=False, drop_last=True)
Settings for distributed training. Create a sampler for data distribution to ensure that different
processes load different data. ###

val_loader = DataLoader(val_set, batch_size=batch, shuffle=False)

lr = args.lr * gpus_per_node
max_epoch = args.epochs
model = ResNet(Block).cuda() if args.enable_gpu else ResNet(Block)

Settings for distributed training. Build a DistributedDataParallel model.
model = nn.parallel.DistributedDataParallel(model)
Settings for distributed training. Build a DistributedDataParallel model.

optimizer = optim.Adam(model.parameters(), lr=lr)
loss_func = torch.nn.CrossEntropyLoss()

os.makedirs(args.output_dir, exist_ok=True)

for epoch in range(1, max_epoch + 1):
 model.train()
 train_loss = 0

 ### Settings for distributed training. DistributedDataParallel sampler. Random numbers are set
 for the DistributedDataParallel sampler based on the current epoch number to avoid loading
 duplicate data. ###
 tr_sampler.set_epoch(epoch)
 ### Settings for distributed training. DistributedDataParallel sampler. Random numbers are set
 for the DistributedDataParallel sampler based on the current epoch number to avoid loading
 duplicate data. ###

 for step, (tr_x, tr_y) in enumerate(tr_loader):
 if args.enable_gpu:
 tr_x, tr_y = tr_x.cuda(), tr_y.cuda()
 out = model(tr_x)
 loss = loss_func(out, tr_y)
 optimizer.zero_grad()
 loss.backward()
 optimizer.step()
 train_loss += loss.item()
 print('train | epoch: %d | loss: %.4f' % (epoch, train_loss / len(tr_loader)))

 val_loss = 0
 pred_record = []
 real_record = []
 model.eval()
 with torch.no_grad():
 for step, (val_x, val_y) in enumerate(val_loader):
 if args.enable_gpu:
 val_x, val_y = val_x.cuda(), val_y.cuda()
 out = model(val_x)
 pred_record += list(np.argmax(out.cpu().numpy(), axis=1))
 real_record += list(val_y.cpu().numpy())
 val_loss += loss_func(out, val_y).item()
 val_accu = accuracy_score(real_record, pred_record)
 print('val | epoch: %d | loss: %.4f | accuracy: %.4f' % (epoch, val_loss / len(val_loader), val_accu),
 '\n')

 if args.rank == 0:
```



```
save ckpt every epoch
torch.save(model.state_dict(), os.path.join(args.output_dir, f'epoch_{epoch}.pth'))

if __name__ == '__main__':
 main()
```

- **Result comparison**

100-epoch **cifar-10** dataset training is completed using two resource types respectively: single-node single-PU and two-node 16-PU. The training duration and test set accuracy are as follows.

**Table 7-29** Training result comparison

| Resource Type | Single-Node Single-PU | Two-Node 16-PU |
|---------------|-----------------------|----------------|
| Duration      | 60 minutes            | 20 minutes     |
| Accuracy      | 80+                   | 80+            |

## Code Example of Distributed Training

The following provides a complete code sample of distributed parallel training for the classification task of ResNet18 on the CIFAR-10 dataset.

The content of the training boot file **main.py** is as follows (if you need to execute a single-node and single-PU training job, delete the code for distributed reconstruction):

```
import datetime
import inspect
import os
import pickle
import random
import logging

import argparse
import numpy as np
from sklearn.metrics import accuracy_score
import torch
from torch import nn, optim
import torch.distributed as dist
from torch.utils.data import TensorDataset, DataLoader
from torch.utils.data.distributed import DistributedSampler

file_dir = os.path.dirname(inspect.getframeinfo(inspect.currentframe()).filename)

def load_pickle_data(path):
 with open(path, 'rb') as file:
 data = pickle.load(file, encoding='bytes')
 return data

def _load_data(file_path):
 raw_data = load_pickle_data(file_path)
 labels = raw_data[b'labels']
 data = raw_data[b'data']
 filenames = raw_data[b'filenames']

 data = data.reshape(10000, 3, 32, 32) / 255
 return data, labels, filenames
```

```
def load_cifar_data(root_path):
 train_root_path = os.path.join(root_path, 'cifar-10-batches-py/data_batch_')
 train_data_record = []
 train_labels = []
 train_filenames = []
 for i in range(1, 6):
 train_file_path = train_root_path + str(i)
 data, labels, filenames = _load_data(train_file_path)
 train_data_record.append(data)
 train_labels += labels
 train_filenames += filenames
 train_data = np.concatenate(train_data_record, axis=0)
 train_labels = np.array(train_labels)

 val_file_path = os.path.join(root_path, 'cifar-10-batches-py/test_batch')
 val_data, val_labels, val_filenames = _load_data(val_file_path)
 val_labels = np.array(val_labels)

 tr_data = torch.from_numpy(train_data).float()
 tr_labels = torch.from_numpy(train_labels).long()
 val_data = torch.from_numpy(val_data).float()
 val_labels = torch.from_numpy(val_labels).long()
 return tr_data, tr_labels, val_data, val_labels

def get_data(root_path, custom_data=False):
 if custom_data:
 train_samples, test_samples, img_size = 5000, 1000, 32
 tr_label = [1] * int(train_samples / 2) + [0] * int(train_samples / 2)
 val_label = [1] * int(test_samples / 2) + [0] * int(test_samples / 2)
 random.seed(2021)
 random.shuffle(tr_label)
 random.shuffle(val_label)
 tr_data, tr_labels = torch.randn((train_samples, 3, img_size, img_size)).float(),
 torch.tensor(tr_label).long()
 val_data, val_labels = torch.randn((test_samples, 3, img_size, img_size)).float(), torch.tensor(
 val_label).long()
 tr_set = TensorDataset(tr_data, tr_labels)
 val_set = TensorDataset(val_data, val_labels)
 return tr_set, val_set
 elif os.path.exists(os.path.join(root_path, 'cifar-10-batches-py')):
 tr_data, tr_labels, val_data, val_labels = load_cifar_data(root_path)
 tr_set = TensorDataset(tr_data, tr_labels)
 val_set = TensorDataset(val_data, val_labels)
 return tr_set, val_set
 else:
 try:
 import torchvision
 from torchvision import transforms
 tr_set = torchvision.datasets.CIFAR10(root='./data', train=True,
 download=True, transform=transforms)
 val_set = torchvision.datasets.CIFAR10(root='./data', train=False,
 download=True, transform=transforms)
 return tr_set, val_set
 except Exception as e:
 raise Exception(
 f"{e}, you can download and unzip cifar-10 dataset manually, "
 "the data url is http://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz")

class Block(nn.Module):

 def __init__(self, in_channels, out_channels, stride=1):
 super().__init__()
 self.residual_function = nn.Sequential(
 nn.Conv2d(in_channels, out_channels, kernel_size=3, stride=stride, padding=1, bias=False),
 nn.BatchNorm2d(out_channels),
```

```
 nn.ReLU(inplace=True),
 nn.Conv2d(out_channels, out_channels, kernel_size=3, padding=1, bias=False),
 nn.BatchNorm2d(out_channels)
)

 self.shortcut = nn.Sequential()
 if stride != 1 or in_channels != out_channels:
 self.shortcut = nn.Sequential(
 nn.Conv2d(in_channels, out_channels, kernel_size=1, stride=stride, bias=False),
 nn.BatchNorm2d(out_channels)
)

 def forward(self, x):
 out = self.residual_function(x) + self.shortcut(x)
 return nn.ReLU(inplace=True)(out)

class ResNet(nn.Module):

 def __init__(self, block, num_classes=10):
 super().__init__()
 self.conv1 = nn.Sequential(
 nn.Conv2d(3, 64, kernel_size=3, padding=1, bias=False),
 nn.BatchNorm2d(64),
 nn.ReLU(inplace=True))
 self.conv2 = self.make_layer(block, 64, 64, 2, 1)
 self.conv3 = self.make_layer(block, 64, 128, 2, 2)
 self.conv4 = self.make_layer(block, 128, 256, 2, 2)
 self.conv5 = self.make_layer(block, 256, 512, 2, 2)
 self.avg_pool = nn.AdaptiveAvgPool2d((1, 1))
 self.dense_layer = nn.Linear(512, num_classes)

 def make_layer(self, block, in_channels, out_channels, num_blocks, stride):
 strides = [stride] + [1] * (num_blocks - 1)
 layers = []
 for stride in strides:
 layers.append(block(in_channels, out_channels, stride))
 in_channels = out_channels
 return nn.Sequential(*layers)

 def forward(self, x):
 out = self.conv1(x)
 out = self.conv2(out)
 out = self.conv3(out)
 out = self.conv4(out)
 out = self.conv5(out)
 out = self.avg_pool(out)
 out = out.view(out.size(0), -1)
 out = self.dense_layer(out)
 return out

 def setup_seed(seed):
 torch.manual_seed(seed)
 torch.cuda.manual_seed_all(seed)
 np.random.seed(seed)
 random.seed(seed)
 torch.backends.cudnn.deterministic = True

 def obs_transfer(src_path, dst_path):
 import moxing as mox
 mox.file.copy_parallel(src_path, dst_path)
 logging.info(f"end copy data from {src_path} to {dst_path}")

 def main():
 seed = datetime.datetime.now().year
 setup_seed(seed)
```

```
parser = argparse.ArgumentParser(description='Pytorch distribute training',
 formatter_class=argparse.ArgumentDefaultsHelpFormatter)
parser.add_argument('--enable_gpu', default='true')
parser.add_argument('--lr', default='0.01', help='learning rate')
parser.add_argument('--epochs', default='100', help='training iteration')

parser.add_argument('--init_method', default=None, help='tcp_port')
parser.add_argument('--rank', type=int, default=0, help='index of current task')
parser.add_argument('--world_size', type=int, default=1, help='total number of tasks')

parser.add_argument('--custom_data', default='false')
parser.add_argument('--data_url', type=str, default=os.path.join(file_dir, 'input_dir'))
parser.add_argument('--output_dir', type=str, default=os.path.join(file_dir, 'output_dir'))
args, unknown = parser.parse_known_args()

args.enable_gpu = args.enable_gpu == 'true'
args.custom_data = args.custom_data == 'true'
args.lr = float(args.lr)
args.epochs = int(args.epochs)

if args.custom_data:
 logging.warning('you are training on custom random dataset, '
 'validation accuracy may range from 0.4 to 0.6.')

Settings for distributed training. Initialize DistributedDataParallel process. The init_method, rank,
and world_size parameters are automatically input by the platform. ###
dist.init_process_group(init_method=args.init_method, backend="nccl", world_size=args.world_size,
rank=args.rank)
Settings for distributed training. Initialize DistributedDataParallel process. The init_method, rank,
and world_size parameters are automatically input by the platform. ###

tr_set, val_set = get_data(args.data_url, custom_data=args.custom_data)

batch_per_gpu = 128
gpus_per_node = torch.cuda.device_count() if args.enable_gpu else 1
batch = batch_per_gpu * gpus_per_node

tr_loader = DataLoader(tr_set, batch_size=batch, shuffle=False)

Settings for distributed training. Create a sampler for data distribution to ensure that different
processes load different data. ###
tr_sampler = DistributedSampler(tr_set, num_replicas=args.world_size, rank=args.rank)
tr_loader = DataLoader(tr_set, batch_size=batch, sampler=tr_sampler, shuffle=False, drop_last=True)
Settings for distributed training. Create a sampler for data distribution to ensure that different
processes load different data. ###

val_loader = DataLoader(val_set, batch_size=batch, shuffle=False)

lr = args.lr * gpus_per_node * args.world_size
max_epoch = args.epochs
model = ResNet(Block).cuda() if args.enable_gpu else ResNet(Block)

Settings for distributed training. Build a DistributedDataParallel model.
model = nn.parallel.DistributedDataParallel(model)
Settings for distributed training. Build a DistributedDataParallel model.

optimizer = optim.Adam(model.parameters(), lr=lr)
loss_func = torch.nn.CrossEntropyLoss()

os.makedirs(args.output_dir, exist_ok=True)

for epoch in range(1, max_epoch + 1):
 model.train()
 train_loss = 0

 ### Settings for distributed training. DistributedDataParallel sampler. Random numbers are set for
 the DistributedDataParallel sampler based on the current epoch number to avoid loading duplicate data.
 ###
```

```
tr_sampler.set_epoch(epoch)
Settings for distributed training. DistributedDataParallel sampler. Random numbers are set for
the DistributedDataParallel sampler based on the current epoch number to avoid loading duplicate data.
###

for step, (tr_x, tr_y) in enumerate(tr_loader):
 if args.enable_gpu:
 tr_x, tr_y = tr_x.cuda(), tr_y.cuda()
 out = model(tr_x)
 loss = loss_func(out, tr_y)
 optimizer.zero_grad()
 loss.backward()
 optimizer.step()
 train_loss += loss.item()
print('train | epoch: %d | loss: %.4f' % (epoch, train_loss / len(tr_loader)))

val_loss = 0
pred_record = []
real_record = []
model.eval()
with torch.no_grad():
 for step, (val_x, val_y) in enumerate(val_loader):
 if args.enable_gpu:
 val_x, val_y = val_x.cuda(), val_y.cuda()
 out = model(val_x)
 pred_record += list(np.argmax(out.cpu().numpy(), axis=1))
 real_record += list(val_y.cpu().numpy())
 val_loss += loss_func(out, val_y).item()
val_accu = accuracy_score(real_record, pred_record)
print('val | epoch: %d | loss: %.4f | accuracy: %.4f' % (epoch, val_loss / len(val_loader), val_accu), '\n')

if args.rank == 0:
 # save ckpt every epoch
 torch.save(model.state_dict(), os.path.join(args.output_dir, f'epoch_{epoch}.pth'))

if __name__ == '__main__':
 main()
```

## FAQs

### 1. How Do I Use Different Datasets in the Sample Code?

- To use the CIFAR-10 dataset in the preceding code, download and decompress the dataset and upload it to the OBS bucket. The file directory structure is as follows:

```
DDP
|--- main.py
|--- input_dir
|----- cifar-10-batches-py
|----- data_batch_1
|----- data_batch_2
|----- ...
```

**DDP** is the code directory specified during training job creation, **main.py** is the preceding code example (the boot file specified during training job creation), and **cifar-10-batches-py** is the unzipped dataset folder (stored in **input\_dir**).

- To use user-defined random data, change the value of **custom\_data** in the code example to **true**.

```
parser.add_argument('--custom_data', default='true')
```

Then, run **main.py**. The parameters for creating a training job are the same as those shown in the preceding figure.

### 2. Why Can I Leave the IP Address of the Master Node Blank for DDP?

The **init method** parameter in **parser.add\_argument('--init\_method', default=None, help='tcp\_port')** contains the IP address and port number of the master node, which are automatically input by the platform.

## 7.7.4 Example: Creating a DDP Distributed Training Job (PyTorch + GPU)

PyTorch's DDP enables efficient distributed training. This section describes how to start DDP training with sample code provided.

- Use PyTorch preset images and run the **mp.spawn** command.
- Use custom images.
  - Run the **torch.distributed.launch** command.
  - Run the **torch.distributed.run** command.

These methods offer adaptable solutions tailored to specific needs and setups.

After reviewing this section, choose the right boot mode for your situation to start using PyTorch distributed training efficiently.

### Creating a Training Job

- Method 1: Use the preset PyTorch framework and run the **mp.spawn** command to start a training job.

For details about parameters for creating a training job, see [Table 7-30](#).

**Table 7-30** Creating a training job (preset image)

| Parameter      | Description                                                                                                                                                                                                    |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Algorithm Type | Select <b>Custom algorithm</b> .                                                                                                                                                                               |
| Boot Mode      | Select <b>Preset image</b> then <b>PyTorch</b> . Select a version as needed.                                                                                                                                   |
| Code Directory | Select the training code path from your OBS bucket, for example, <b>obs://test-modelarts/code/</b> .                                                                                                           |
| Boot File      | Select the Python boot script of the training job in the code directory, for example, <b>obs://test-modelarts/code/main.py</b> .                                                                               |
| Hyperparameter | To use a single-node multi-PU flavor, set the hyperparameters <b>world_size</b> and <b>rank</b> .<br>If you choose a flavor with multiple instances, these hyperparameters are automatically set by ModelArts. |

- Method 2: Use a custom image and run the **torch.distributed.launch** command to start a training job.

For details about parameters for creating a training job, see [Table 7-31](#).

**Table 7-31** Creating a training job (custom image + `torch.distributed.launch`)

| Parameter      | Description                                                                                          |
|----------------|------------------------------------------------------------------------------------------------------|
| Algorithm Type | Select <b>Custom algorithm</b> .                                                                     |
| Boot Mode      | Select <b>Custom image</b> .                                                                         |
| Image          | Select a PyTorch image for training.                                                                 |
| Code Directory | Select the training code path from your OBS bucket, for example, <b>obs://test-modelarts/code/</b> . |
| Boot Command   | Enter the Python boot command of the image, for example:<br>bash \${MA_JOB_DIR}/code/torchlaunch.sh  |

- Method 3: Use a custom image and run the `torch.distributed.run` command to start a training job.

For details about parameters for creating a training job, see [Table 7-32](#).

**Table 7-32** Creating a training job (custom image + `torch.distributed.run`)

| Parameter      | Description                                                                                          |
|----------------|------------------------------------------------------------------------------------------------------|
| Algorithm Type | Select <b>Custom algorithm</b> .                                                                     |
| Boot Mode      | Select <b>Custom image</b> .                                                                         |
| Image          | Select a PyTorch image for training.                                                                 |
| Code Directory | Select the training code path from your OBS bucket, for example, <b>obs://test-modelarts/code/</b> . |
| Boot Command   | Enter the Python boot command of the image, for example:<br>bash \${MA_JOB_DIR}/code/torchrun.sh     |

## Code Example

Upload the following files to an OBS bucket:

```
code # Root directory of the code
├── torch_ddp.py # Code file for PyTorch DDP training
├── main.py # Boot file for starting training using the PyTorch preset image and the mp.spawn
command
├── torchlaunch.sh # Boot file for starting training using the custom image and the
torch.distributed.launch command
├── torchrun.sh # Boot file for starting training using the custom image and the
torch.distributed.run command
```

### `torch_ddp.py`

```
import os
import torch
import torch.distributed as dist
import torch.nn as nn
```

```
import torch.optim as optim
from torch.nn.parallel import DistributedDataParallel as DDP

Start training by running mp.spawn.
def init_from_arg(local_rank, base_rank, world_size, init_method):
 rank = base_rank + local_rank
 dist.init_process_group("nccl", rank=rank, init_method=init_method, world_size=world_size)
 ddp_train(local_rank)

Start training by running torch.distributed.launch or torch.distributed.run.
def init_from_env():
 dist.init_process_group(backend='nccl', init_method='env://')
 local_rank=int(os.environ["LOCAL_RANK"])
 ddp_train(local_rank)

def cleanup():
 dist.destroy_process_group()

class ToyModel(nn.Module):
 def __init__(self):
 super(ToyModel, self).__init__()
 self.net1 = nn.Linear(10, 10)
 self.relu = nn.ReLU()
 self.net2 = nn.Linear(10, 5)
 def forward(self, x):
 return self.net2(self.relu(self.net1(x)))

def ddp_train(device_id):
 # create model and move it to GPU with id rank
 model = ToyModel().to(device_id)
 ddp_model = DDP(model, device_ids=[device_id])
 loss_fn = nn.MSELoss()
 optimizer = optim.SGD(ddp_model.parameters(), lr=0.001)
 optimizer.zero_grad()
 outputs = ddp_model(torch.randn(20, 10))
 labels = torch.randn(20, 5).to(device_id)
 loss_fn(outputs, labels).backward()
 optimizer.step()
 cleanup()

if __name__ == "__main__":
 init_from_env()
```

### main.py

```
import argparse
import torch
import torch.multiprocessing as mp

parser = argparse.ArgumentParser(description='ddp demo args')
parser.add_argument('--world_size', type=int, required=True)
parser.add_argument('--rank', type=int, required=True)
parser.add_argument('--init_method', type=str, required=True)
args, unknown = parser.parse_known_args()

if __name__ == "__main__":
 n_gpus = torch.cuda.device_count()
 world_size = n_gpus * args.world_size
 base_rank = n_gpus * args.rank
 # Call the start function in the DDP sample code.
 from torch_ddp import init_from_arg
 mp.spawn(init_from_arg,
 args=(base_rank, world_size, args.init_method),
 nprocs=n_gpus,
 join=True)
```

### torchlaunch.sh

```
#!/bin/bash
Default system environment variables. Do not modify them.
MASTER_HOST="$VC_WORKER_HOSTS"
```



```
MASTER_ADDR="{VC_WORKER_HOSTS%%,*}"
MASTER_PORT="6060"
JOB_ID="1234"
NNODES="$MA_NUM_HOSTS"
NODE_RANK="$VC_TASK_INDEX"
NGPUS_PER_NODE="$MA_NUM_GPUS"

Custom environment variables to specify the Python script and parameters.
PYTHON_SCRIPT=${MA_JOB_DIR}/code/torch_ddp.py
PYTHON_ARGS=""

CMD="python -m torch.distributed.launch \
 --nnodes=$NNODES \
 --node_rank=$NODE_RANK \
 --nproc_per_node=$NGPUS_PER_NODE \
 --master_addr $MASTER_ADDR \
 --master_port=$MASTER_PORT \
 --use_env \
 $PYTHON_SCRIPT \
 $PYTHON_ARGS"

echo $CMD
$CMD
```

### torchrun.sh



#### WARNING

In PyTorch 2.1, you must set **rdzv\_backend** to **static**: **--rdzv\_backend=static**.

```
#!/bin/bash
Default system environment variables. Do not modify them.
MASTER_HOST="$VC_WORKER_HOSTS"
MASTER_ADDR="{VC_WORKER_HOSTS%%,*}"
MASTER_PORT="6060"
JOB_ID="1234"
NNODES="$MA_NUM_HOSTS"
NODE_RANK="$VC_TASK_INDEX"
NGPUS_PER_NODE="$MA_NUM_GPUS"

Custom environment variables to specify the Python script and parameters.
PYTHON_SCRIPT=${MA_JOB_DIR}/code/torch_ddp.py
PYTHON_ARGS=""

if [[$NODE_RANK == 0]]; then
 EXT_ARGS="--rdzv_conf=is_host=1"
else
 EXT_ARGS=""
fi

CMD="python -m torch.distributed.run \
 --nnodes=$NNODES \
 --node_rank=$NODE_RANK \
 $EXT_ARGS \
 --nproc_per_node=$NGPUS_PER_NODE \
 --rdzv_id=$JOB_ID \
 --rdzv_backend=c10d \
 --rdzv_endpoint=$MASTER_ADDR:$MASTER_PORT \
 $PYTHON_SCRIPT \
 $PYTHON_ARGS"

echo $CMD
$CMD
```

## 7.7.5 Example: Creating a DDP Distributed Training Job (PyTorch + NPU)

PyTorch's DDP enables efficient distributed training on Ascend accelerator cards. Implementing this with custom images and boot commands can pose challenges.

This guide offers a solution: configure custom images and boot commands to run PyTorch DDP training on Ascend accelerator cards. It supports user-specific needs and adapts to various training scenarios.

With this approach, users can streamline PyTorch DDP training for better performance on Ascend accelerator cards.

### Prerequisites

An Ascend accelerator card resource pool is available.

### Creating a Training Job

The following table describes the parameters you need to configure during training job creation.

**Table 7-33** Parameters for creating a training job

| Parameter      | Description                                                                                                                                                                                                  |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Algorithm Type | Select <b>Custom algorithm</b> .                                                                                                                                                                             |
| Boot Mode      | Select <b>Custom image</b> .                                                                                                                                                                                 |
| Image          | Select a custom image for training.                                                                                                                                                                          |
| Code Directory | Select the code directory required for this training job, for example, <b>obs://test-modelarts/ascend/code/</b> in this case.                                                                                |
| Boot Command   | Python boot command of the image, for example, <b>bash \${MA_JOB_DIR}/code/run_torch_ddp_npu.sh</b> in this case. For details about the complete code of the boot script, see <a href="#">Code Example</a> . |

### (Optional) Enabling Ranktable Dynamic Routing

To use ranktable dynamic routing for network acceleration, contact technical support to enable cabinet scheduling permission. For details, see [Enabling Dynamic Route Acceleration for Training Jobs](#).

### Code Example

The following shows an example boot script of a training job.

 NOTE

To store the generated plog data, you need to specify the path in the startup script as **/home/ma-user/modelarts/log/modelarts-job-*id*/worker-*index***/. The system will automatically upload the \*.log file in the **/home/ma-user/modelarts/log/** directory to the OBS log directory of your training job. The system will only upload the log files (larger than 0 MB) in the local directory to the corresponding parent directory. Unlike MindSpore, PyTorch NPU plog logs are organized by worker instead of rank ID. PyTorch NPU does not rely on the rank table file.

```
#!/bin/bash

MA preset envs
MASTER_HOST="$VC_WORKER_HOSTS"
MASTER_ADDR="{VC_WORKER_HOSTS%%,*}"
NNODES="$MA_NUM_HOSTS"
NODE_RANK="$VC_TASK_INDEX"
also indicates NPU per node
NGPUS_PER_NODE="$MA_NUM_GPUS"

self-define, it can be changed to >=10000 port
MASTER_PORT="38888"

replace ${MA_JOB_DIR}/code/torch_ddp.py to the actual training script
PYTHON_SCRIPT=${MA_JOB_DIR}/code/torch_ddp.py
PYTHON_ARGS=""

export HCCL_WHITELIST_DISABLE=1

set npu plog env
ma_vj_name=`echo ${MA_VJ_NAME} | sed 's:ma-job:modelarts-job:g`
task_name="worker-${VC_TASK_INDEX}"
task_plog_path=${MA_LOG_DIR}/${ma_vj_name}/${task_name}

mkdir -p ${task_plog_path}
export ASCEND_PROCESS_LOG_PATH=${task_plog_path}

echo "plog path: ${ASCEND_PROCESS_LOG_PATH}"

set hccl timeout time in seconds
export HCCL_CONNECT_TIMEOUT=1800

replace ${ANACONDA_DIR}/envs/${ENV_NAME}/bin/python to the actual python
CMD="{ANACONDA_DIR}/envs/${ENV_NAME}/bin/python -m torch.distributed.launch \
 --nnodes=$NNODES \
 --node_rank=$NODE_RANK \
 --nproc_per_node=$NGPUS_PER_NODE \
 --master_addr=$MASTER_ADDR \
 --master_port=$MASTER_PORT \
 --use_env \
 $PYTHON_SCRIPT \
 $PYTHON_ARGS"
"
echo $CMD
$CMD
```

## 7.8 Enabling Dynamic Route Acceleration for Training Jobs

Distributed training faces performance issues because networks struggle with slow data transfer and poor bandwidth usage when exchanging information across multiple nodes. To address these challenges, ModelArts offers dynamic routing acceleration. It smartly optimizes network paths for training jobs, boosting overall performance. This guide explains how to enable this feature on ModelArts,

covering both preset frameworks and custom images. It also includes setup instructions and tips to maximize your distributed training results.

## Notes and Constraints

- Dynamic routing acceleration can only be enabled in the following training scenarios:
  - [Scenario 1: Using the Ascend-Powered-Engine Preset Image, MindSpore, and NPUs for Training](#)
  - [Scenario 2: Using a Custom Image, PyTorch, and NPUs for Training](#)
- The training job must use Python 3.7, 3.8, or 3.9.
- Before enabling dynamic routing acceleration, contact ModelArts technical support to ensure that the cabinet plug-in and scheduling permissions of the cluster are enabled.

## Scenario 1: Using the Ascend-Powered-Engine Preset Image, MindSpore, and NPUs for Training

When using the Ascend-Powered-Engine preset image to create a training job, refer to [Table 7-34](#) to create a training job and enable dynamic routing acceleration. The table below describes only key parameters. Configure other parameters based on actual needs.

**Table 7-34** Using a preset image to create a training job

| Step                 | Parameter            | Description                                                                                                                                                                                                                      |
|----------------------|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Environment settings | Algorithm Type       | Select <b>Custom algorithm</b> .                                                                                                                                                                                                 |
|                      | Boot Mode            | Select <b>Preset image</b> .                                                                                                                                                                                                     |
|                      | Engine and Version   | Select <b>Ascend-Powered-Engine</b> and a MindSpore-related engine version.                                                                                                                                                      |
|                      | Code Directory       | Select the OBS directory where the training code file is stored.<br><br>Dynamic routing acceleration improves network communication by adjusting the rank ID. To prevent communication issues, unify the rank usage in the code. |
|                      | Boot File            | Select the Python boot script of the training job in the code directory,                                                                                                                                                         |
| Training settings    | Environment Variable | Add the following environment variables:<br>ROUTE_PLAN = true<br><br>Do not configure the environment variable <b>MA_RUN_METHOD</b> . Ensure that the boot file of the training job is started using the rank table file.        |

| Step              | Parameter      | Description                                                                                                                                                                                                                                                                                                                                                                                 |
|-------------------|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Resource settings | Resource Pool  | Select a dedicated resource pool.                                                                                                                                                                                                                                                                                                                                                           |
|                   | Specifications | Select instance specifications that meet the following requirements: <ul style="list-style-type: none"><li>• All processing units (PUs) on each node must be fully utilized. Otherwise, the effectiveness of dynamic routing acceleration may be impacted. For example, if a node has eight PUs, all eight must be used.</li><li>• The resources must be Ascend Snt9b or Snt9b23.</li></ul> |
|                   | Compute Nodes  | Select at least three compute nodes.                                                                                                                                                                                                                                                                                                                                                        |

## Scenario 2: Using a Custom Image, PyTorch, and NPUs for Training

When using a custom image and Ascend resource pool to create a training job, refer to [Table 7-35](#) to create a training job and enable dynamic routing acceleration. The table below describes only key parameters. Configure other parameters based on actual needs.

**Table 7-35** Using a custom image to create a training job

| Step                 | Parameter                 | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|----------------------|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Environment settings | Algorithm Type            | Select <b>Custom algorithm</b> .                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|                      | Boot Mode                 | Select <b>Custom image</b> .                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|                      | Image                     | Select a custom image for training. The training image must use the PyTorch framework.                                                                                                                                                                                                                                                                                                                                                                                                     |
|                      | Code Directory (Optional) | Select the OBS directory where the training code file is stored.<br><br>Dynamic routing acceleration improves network communication by adjusting the rank ID. To prevent communication issues, unify the rank usage in the code.                                                                                                                                                                                                                                                           |
|                      | Boot Command              | Enter the Python boot command of the image.<br>Modify the following code in the training boot script. The values vary according to the NPU hardware. <ul style="list-style-type: none"><li>• Snt9b scenario<br/>MASTER_ADDR="\${MA_VJ_NAME}-\${MA_TASK_NAME}-\${MA_MASTER_INDEX:-0}.\${MA_VJ_NAME}"<br/>NODE_RANK="\${RANK_AFTER_ACC:-\$VC_TASK_INDEX}"</li><li>• Snt9b23 scenario<br/>MASTER_ADDR="\${VC_WORKER_HOSTS%%,*}"<br/>NODE_RANK="\${RANK_AFTER_ACC:-\$VC_TASK_INDEX}"</li></ul> |

| Step              | Parameter            | Description                                                                                                                                                                                                                                                                                                                                                                             |
|-------------------|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Training settings | Environment Variable | Add the following environment variables:<br>ROUTE_PLAN = true                                                                                                                                                                                                                                                                                                                           |
| Resource settings | Resource Pool        | Select a dedicated resource pool.                                                                                                                                                                                                                                                                                                                                                       |
|                   | Specifications       | Select instance specifications that meet the following requirements: <ul style="list-style-type: none"><li>All processing units (PUs) on each node must be fully utilized. Otherwise, the effectiveness of dynamic routing acceleration may be impacted. For example, if a node has eight PUs, all eight must be used.</li><li>The resources must be Ascend Snt9b or Snt9b23.</li></ul> |
|                   | Compute Nodes        | Select at least three compute nodes.                                                                                                                                                                                                                                                                                                                                                    |

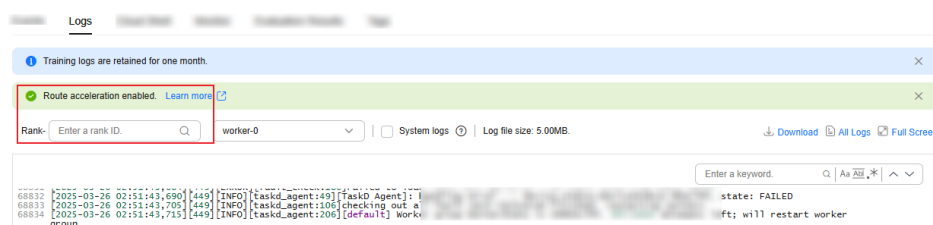
## Viewing Training Logs of Dynamic Route Acceleration

When using an Ascend resource pool for training, you can check whether the route is enabled and query the log information of each rank in the **Logs** tab of the training job details page.

- Log in to the [ModelArts management console](#).
- In the navigation pane on the left, choose **Model Training > Training Jobs**.
- In the training job list, click the target job to access its details page.
- Click the **Logs** tab.

You can view that dynamic routing has been enabled for the training job and search for logs by rank ID.

**Figure 7-16** Viewing dynamic route acceleration logs



## 7.9 Incremental Model Training

### What Is Incremental Training?

Incremental learning is a machine learning method that enables AI models to learn from new data without restarting the training process. It builds on existing

knowledge, allowing the model to expand its capabilities and improve its performance over time.

Incremental learning allows for training on data in smaller chunks, reducing storage needs and alleviating resource constraints. It also conserves computing power and time, and lowers retraining costs.

Incremental training is ideal for these scenarios:

- Continuous data updates: It allows models to adapt to new data without retraining.
- Resource constraints: It is a more economical choice when retraining a model is too costly.
- Avoiding knowledge loss: It retains old knowledge while learning new information, preventing the model from forgetting what it has learned.

Incremental training is used in various fields, including natural language processing, computer vision, and recommendation systems. It makes AI systems more flexible and adaptable, allowing them to handle changing data in real-world environments.

## Implementing Incremental Training in ModelArts Standard

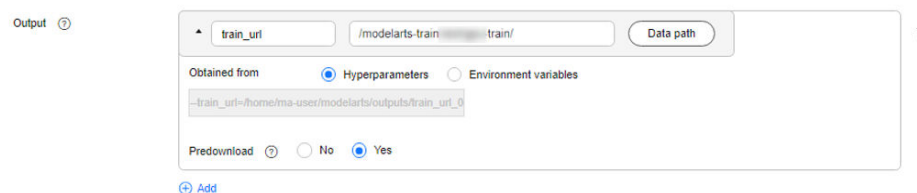
The checkpoint mechanism enables incremental training.

During model training, training results (including but not limited to epochs, model weights, optimizer status, and scheduler status) are continuously saved. To add data and resume a training job, load a checkpoint and use the checkpoint information to initialize the training status. To do so, add **reload ckpt** to the code.

To incrementally train a model in ModelArts, configure the training output.

When creating a training job, set the data path to the training output, save checkpoints in this data path, and set **Predownload** to **Yes**. If you set **Predownload** to **Yes**, the system automatically downloads the **checkpoint** file in the training output data path to a local directory of the training container before the training job is started.

**Figure 7-17** Configuring training output



### reload ckpt for PyTorch

- Use either of the following methods to save a PyTorch model.
  - Save model parameters only.

```
state_dict = model.state_dict()
torch.save(state_dict, path)
```
  - Save the entire model (not recommended).

```
torch.save(model, path)
```

- Save the data generated during model training at regular intervals based on steps and time.

The data includes the network weight, optimizer weight, and epoch, which will be used to resume the interrupted training.

```
checkpoint = {
 "net": model.state_dict(),
 "optimizer": optimizer.state_dict(),
 "epoch": epoch
}
if not os.path.isdir('model_save_dir'):
 os.makedirs('model_save_dir')
torch.save(checkpoint, 'model_save_dir/ckpt_{}.pth'.format(str(epoch)))
```

- Check the complete code example below.

```
import os
import argparse
parser = argparse.ArgumentParser()
parser.add_argument("--train_url", type=str)
args, unparsed = parser.parse_known_args()
args = parser.parse_known_args()
train_url is set to /home/ma-user/modelarts/outputs/train_url_0.
train_url = args.train_url

Check whether there is a model file in the output path. If there is no file, the model will be trained
from the beginning by default. If there is a model file, the CKPT file with the maximum epoch value
will be loaded as the pre-trained model.
if os.listdir(train_url):
 print('> load last ckpt and continue training!!')
 last_ckpt = sorted([file for file in os.listdir(train_url) if file.endswith(".pth")])[-1]
 local_ckpt_file = os.path.join(train_url, last_ckpt)
 print('last_ckpt:', last_ckpt)
 # Load the checkpoint.
 checkpoint = torch.load(local_ckpt_file)
 # Load the parameters that can be learned by the model.
 model.load_state_dict(checkpoint['net'])
 # Load optimizer parameters.
 optimizer.load_state_dict(checkpoint['optimizer'])
 # Obtain the saved epoch. The model will continue to be trained based on the epoch value.
 start_epoch = checkpoint['epoch']
start = datetime.now()
total_step = len(train_loader)
for epoch in range(start_epoch + 1, args.epochs):
 for i, (images, labels) in enumerate(train_loader):
 images = images.cuda(non_blocking=True)
 labels = labels.cuda(non_blocking=True)
 # Forward pass
 outputs = model(images)
 loss = criterion(outputs, labels)
 # Backward and optimize
 optimizer.zero_grad()
 loss.backward()
 optimizer.step()
 ...

 # Save the network weight, optimizer weight, and epoch during model training.
 checkpoint = {
 "net": model.state_dict(),
 "optimizer": optimizer.state_dict(),
 "epoch": epoch
 }
 if not os.path.isdir(train_url):
 os.makedirs(train_url)
 torch.save(checkpoint, os.path.join(train_url, 'ckpt_best_{}.pth'.format(epoch)))
```

## 7.10 Automatic Model Tuning (AutoSearch)



## 7.10.1 Overview

ModelArts automatically searches for optimal hyperparameters for your models, saving time and effort.

During training, hyperparameters like **learning\_rate** and **weight\_decay** need to be adjusted. ModelArts hyperparameter search optimizes these settings automatically, outperforming manual tuning in speed and precision.

ModelArts supports the following hyperparameter search algorithms:

- Bayesian Optimization (SMAC)
- Tree-structured Parzen Estimator (TPE)
- Simulated Annealing

### Bayesian Optimization (SMAC)

Bayesian optimization assumes a functional relationship between hyperparameters and the objective function. It estimates the mean and variance of objective function values at other search points using Gaussian process regression, based on the evaluation values of the searched hyperparameters. The mean and variance are then used to construct the acquisition function, which identifies the next search point as its maximum value. Bayesian optimization reduces the number of iterations and search time by leveraging previous evaluation results, but it can struggle to find the global optimal solution.

**Table 7-36** Bayesian optimization parameters

| Parameter   | Description                                                                                              | Recommended Value                                                                                                           |
|-------------|----------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| num_samples | Number of hyperparameter groups to search                                                                | This integer ranges from 10 to 20. Larger values increase search time but improve results.                                  |
| kind        | Acquisition function type                                                                                | This string defaults to <b>ucb</b> . Other options are <b>ei</b> and <b>poi</b> , but it is best to stick with the default. |
| kappa       | Adjustment parameter for the <b>ucb</b> acquisition function, representing the upper confidence boundary | This float value should remain unchanged.                                                                                   |
| xi          | Adjustment parameter for <b>poi</b> and <b>ei</b> acquisition functions.                                 | This float value should remain unchanged.                                                                                   |

### Tree-structured Parzen Estimator (TPE)

The TPE algorithm uses a Gaussian mixture model to learn model hyperparameters. On each trial, TPE fits two Gaussian mixture models: one to the best objective values and another to the remaining values. It selects the hyperparameter value that maximizes the ratio of these two models.

**Table 7-37** TPE parameters

| Parameter        | Description                                                                                           | Recommended Value                                                                          |
|------------------|-------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------|
| num_samples      | Number of hyperparameter groups to search                                                             | This integer ranges from 10 to 20. Larger values increase search time but improve results. |
| n_initial_points | Number of random evaluations of the objective function before using tree-structured parzen estimators | This integer should remain unchanged.                                                      |
| gamma            | Quantile used by the TPE algorithm to split $l(x)$ and $g(x)$                                         | This float value ranges from 0 to 1 and should remain unchanged.                           |

## Simulated Annealing

The simulated annealing algorithm is a simple and effective search method that uses the smoothness of the response surface. It starts with a previous trial point and samples each hyperparameter from a distribution similar to the prior, but with a higher concentration around the chosen point. Over time, the algorithm focuses on sampling points closer to the best ones. Occasionally, it may select a runner-up trial as the best to avoid local optima with a certain probability.

**Table 7-38** Simulated annealing parameters

| Parameter    | Description                                                                                     | Recommended Value                                                                          |
|--------------|-------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------|
| num_samples  | Number of hyperparameter groups to search                                                       | This integer ranges from 10 to 20. Larger values increase search time but improve results. |
| avg_best_idx | Mean of the geometric distribution used to select trials for exploration, based on their scores | This float value should remain unchanged.                                                  |
| shrink_coef  | Rate at which the sampling neighborhood size decreases as more points are explored              | This float value should remain unchanged.                                                  |

## 7.10.2 Creating a Training Job for Automatic Model Tuning

### Context

To use ModelArts hyperparameter search, the AI engine must be either **pytorch\_1.8.0-cuda\_10.2-py\_3.7-ubuntu\_18.04-x86\_64** or **tensorflow\_2.1.0-cuda\_10.1-py\_3.7-ubuntu\_18.04-x86\_64**, and the hyperparameter to be optimized must be a float value.

To perform a hyperparameter search without any code modification, follow these steps:

1. [Preparations](#)
2. [Creating an Algorithm](#)
3. [Creating a Training Job](#)
4. [Viewing Details About a Hyperparameter Search Job](#)

## Preparations

- Create a dataset in ModelArts or upload a training dataset to an OBS directory.
- Upload your training script to an OBS directory. For details about how to develop a training script, see [Developing Code for Training Using a Preset Image](#).
- Print search indicator parameters in the training code.
- Create at least one empty folder in OBS for storing training outputs.
- Make sure your account is not in arrears, as training jobs consume resources.
- Make sure your OBS directory and ModelArts are in the same region.

## Creating an Algorithm

Log in to the [ModelArts console](#) and create a custom algorithm by referring to [Creating an Algorithm](#). The image must use the **pytorch\_1.8.0-cuda\_10.2-py\_3.7-ubuntu\_18.04-x86\_64** or **tensorflow\_2.1.0-cuda\_10.1-py\_3.7-ubuntu\_18.04-x86\_64** engine.

To define hyperparameters for optimization, specify the name, type, default value, and constraints in **Hyperparameter**. For details, see [Table 7-15](#).

To enable auto search for the algorithm, select **autoSearch(S)**, print search parameters in the code, and configure the following parameters. ModelArts uses a regular expression to obtain search indicator parameters during an auto search, and then performs hyperparameter optimization based on the specified optimization direction.

Figure 7-18 Enabling auto search

Hyperparameter

| Name | Type  | Default | Required | Description | Operations |
|------|-------|---------|----------|-------------|------------|
| rate | Float |         | Y...     |             | Delete     |

Supported Policies

☒ autoSearch(S)

Search Indicator

| Name | Optimization Direction | Counter regularization |
|------|------------------------|------------------------|
|      | max                    |                        |

Setting Automatic Search Parameters

| Name | Value | Unit |
|------|-------|------|
| rate |       |      |

Search Algorithm Configuration

| Algorithm        | Parameter    | Value |
|------------------|--------------|-------|
| bayes_opt_search | avg_best_idx | 2.0   |
| tpe_search       | shrink_coef  | 0.1   |
| anneal_search    | num_samples  | 20    |

- Search Indicator

The search indicator represents the value of the objective function, such as loss or accuracy. By optimizing and converging this value in the desired direction, you can find the optimal hyperparameters to enhance model accuracy and convergence speed.

**Table 7-39** Search indicator parameters

| Parameter              | Description                                                                                                       |
|------------------------|-------------------------------------------------------------------------------------------------------------------|
| Name                   | Enter a search indicator name. This value must be identical to the search indicator parameter in the code.        |
| Optimization Direction | Select <b>max</b> or <b>min</b> .                                                                                 |
| Counter regularization | Enter a regular expression or click <b>Generate Intelligently</b> to generate a regular expression automatically. |

- Setting Automatic Search Parameters

Select hyperparameters from the **Hyperparameters** configuration. Only float-type hyperparameters are supported. After selecting **autoSearch(S)**, set the value range.

- Search Algorithm Configuration

ModelArts has three built-in algorithms for hyperparameter search. You can select one or more algorithms as needed. The algorithms and their parameter description are as follows:

- bayes\_opt\_search: **Bayesian Optimization (SMAC)**
- tpe\_search: **Tree-structured Parzen Estimator (TPE)**
- anneal\_search: **Simulated Annealing**

After creating the algorithm, use it to create a training job.

## Creating a Training Job

Log in to the [ModelArts console](#) and create a training job by referring to [Creating a Production Training Job](#).

If you select an algorithm that supports hyperparameter search, click the button for range setting to enable hyperparameter search.

Figure 7-19 Enabling hyperparameter search

Hyperparameter

on\_device\_model

=

0

2

⊕ Add

★ Auto Search Metric

rate

★ Auto Search Algorithm

bayes\_opt\_search

tpe\_search

anneal\_search

★ Auto Search Parameters

kind

=

ucb

kappa

=

2.5

xi

=

0.0

num\_samples

=

20

seed

=

1

After enabling the hyperparameter search, you can configure the search indicator, search algorithm, and search algorithm parameters. These parameters have the same values as the hyperparameters of the algorithm you created.

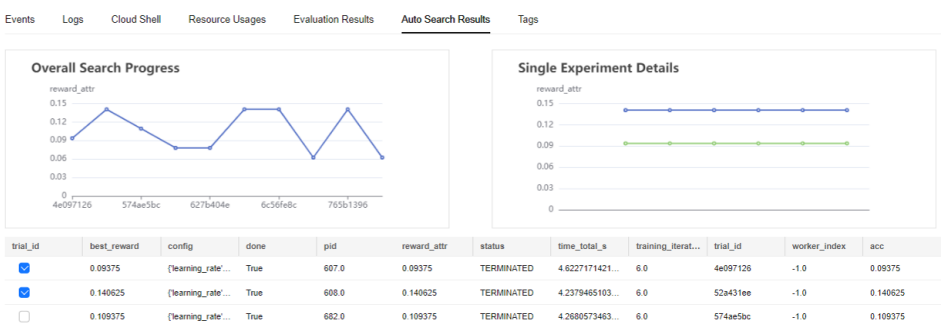
The hyperparameter search job will take some time to run after it is created.

## Viewing Details About a Hyperparameter Search Job

After a training job is completed, you can review the results of the automated hyperparameter search to evaluate the job's performance.

If the training job is a hyperparameter search job, go to the training job details page and click the **Auto Search Results** tab to view the hyperparameter search results.

Figure 7-20 Hyperparameter search results



## 7.11 High Model Training Reliability

### 7.11.1 Training Job Fault Tolerance Check

#### Description

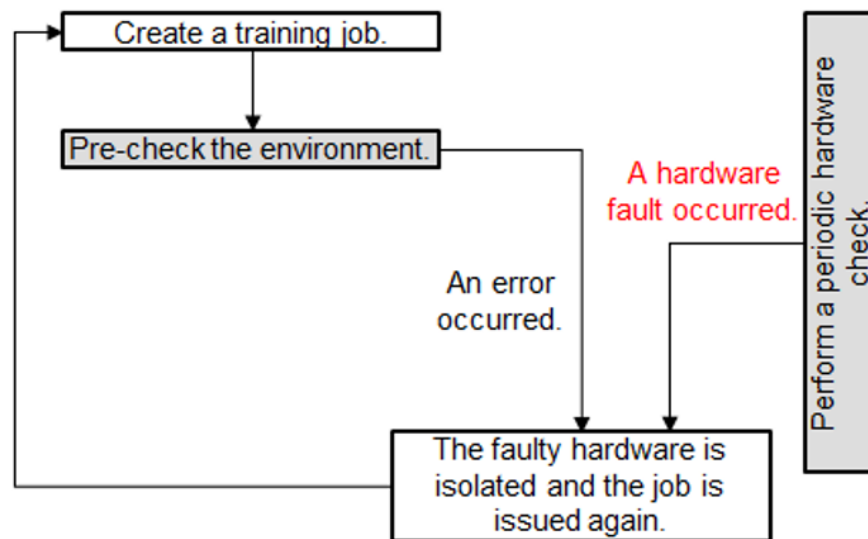
During model training, a training failure may occur due to a hardware fault. For hardware faults, ModelArts provides fault tolerance check to isolate faulty nodes to improve user experience in training.

The fault tolerance check involves environment pre-check and periodic hardware check. If any fault is detected during either of the checks, ModelArts automatically isolates the faulty hardware and issues the training job again. In distributed training, the fault tolerance check will be performed on all compute nodes used by the training job.

The following shows four failure scenarios, among which the failure in scenario 4 is not caused by a hardware fault. You can enable fault tolerance in the other three scenarios to automatically resume the training job.

- Scenario 1: The environment pre-check fails, and the hardware is faulty. Then, ModelArts automatically isolates all faulty nodes and issues the training job again.

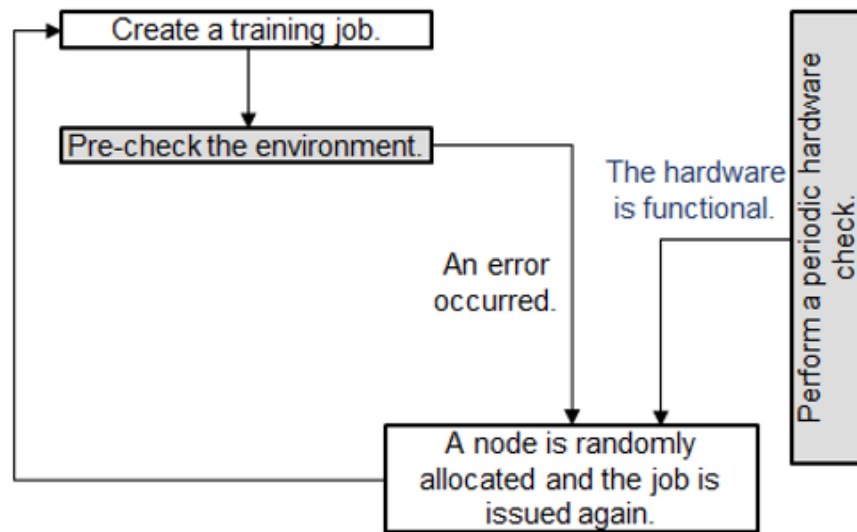
**Figure 7-21** Pre-check failure and hardware fault



Pre-check failure and hardware fault

- Scenario 2: The environment pre-check fails but the hardware is functional. Then, ModelArts randomly allocates nodes and issues the training job again.

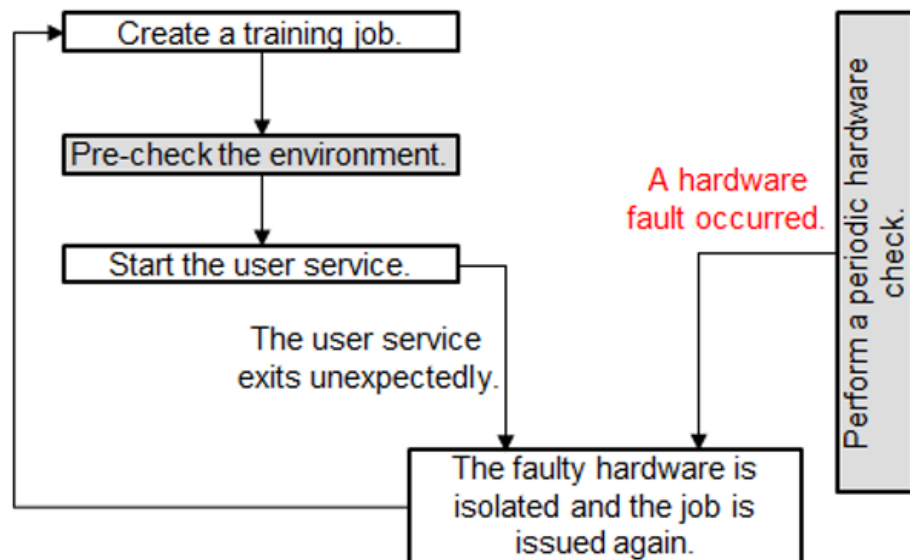
**Figure 7-22** Pre-check failure but functional hardware



### Pre-check failure and normal hardware

- Scenario 3: The environment pre-check is successful and the user service starts. A hardware fault occurs and the user service exits unexpectedly. Then, ModelArts automatically isolates all faulty nodes and issues the training job again.

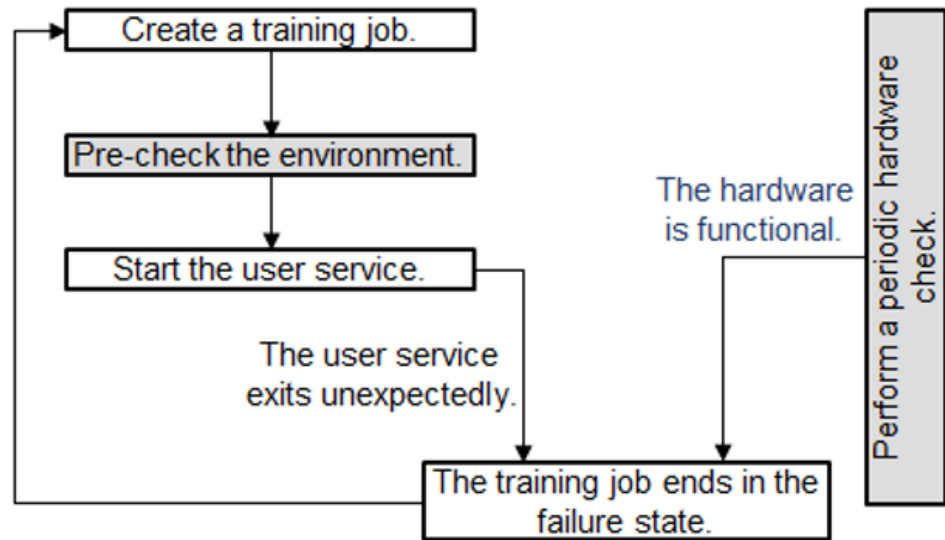
**Figure 7-23** Service failure and hardware fault



### Service failure and hardware fault

- Scenario 4: The environment pre-check is successful and the user service starts. The hardware is functional. A fault occurs in the user service, the training job ends in the failure state.

**Figure 7-24** Service failure and functional hardware



Service failure and normal hardware

After the faulty node is isolated, ModelArts creates a training job on new compute nodes. If the resources provided by the resource pool are limited, the re-issued training job will be queued with the highest priority. If the waiting time exceeds 30 minutes, the training job will automatically exit. This indicates that the resources are so limited that the training job cannot start. In this case, buy a dedicated resource pool to obtain dedicated resources.

If you use a dedicated resource pool to create a training job, the faulty nodes identified during the fault tolerance check will be removed. The system automatically adds healthy compute nodes to the dedicated resource pool. (This function is coming soon.)

More details of a fault tolerance check:

1. [Enabling Fault Tolerance Check](#)
2. [Check Items and Conditions](#)
3. [Effect of a Fault Tolerance Check](#)
4. After the environment pre-check is successful, any hardware fault will interrupt the user service. Add the reload ckpt code logic to the training so that the pre-trained model saved before the training is interrupted can be obtained. For details, see [Resumable Training](#).

## Enabling Fault Tolerance Check

To enable fault tolerance check, enable auto restart when creating a training job.

- Configure fault tolerance check on the ModelArts Standard console:  
Enable **Auto Restart** on the ModelArts management console. **Auto Restart** is disabled by default, indicating that the job will not be re-issued and the environment pre-check will not be enabled. After **Auto Restart** is enabled, the number of restart retries ranges from 1 to 128.



**Figure 7-25** Auto Restart

### Fault tolerance and recovery

#### ☒ Auto Restart

If a training job encounters an exception due to environment is:  
your code logic supports resumable training before enabling thi

Maximum number of restarts ?

— | 3 | +

- Configure fault tolerance check using an API:

Enable auto restart upon a fault using an API. When creating a training job, configure the **fault-tolerance/job-retry-num** field in **annotations** of the **metadata** field.

If the **fault-tolerance/job-retry-num** field is added, auto restart is enabled. The value can be an integer ranging from **1** to **128**, specifying the maximum number of times that a job can be re-issued. If this hyperparameter is not specified, the default value **0** is used, indicating that the job will not be re-issued and the environment pre-check will not be enabled.

**Figure 7-26** Setting the API

```
{
 "kind": "job",
 "metadata": {
 "annotations": {
 "fault-tolerance/job-retry-num": "3"
 }
 }
}
```

## Check Items and Conditions

| Check Item                           | Item (Log Keyword) | Execution Condition | Requirements for a Check                                                                                       |
|--------------------------------------|--------------------|---------------------|----------------------------------------------------------------------------------------------------------------|
| Domain name detection                | dns                | None                | The domain names of the volcano containers in the .host file in <b>/etc/volcano</b> are successfully resolved. |
| Disk size - Container root directory | disk-size root     | None                | The directory is greater than 32 GB.                                                                           |
| Disk size - /dev/shm                 | disk-size shm      | None                | The directory is greater than 1 GB.                                                                            |

| Check Item          | Item (Log Keyword) | Execution Condition                      | Requirements for a Check                                                                                                                                                                          |
|---------------------|--------------------|------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Disk size - / cache | disk-size cache    | None                                     | The directory is greater than 32 GB.                                                                                                                                                              |
| ulimit check        | ulimit             | An IB network is used.                   | <ul style="list-style-type: none"><li>• Maximum locked memory &gt; 16000</li><li>• Open files &gt; 1000000</li><li>• Stack size &gt; 8000</li><li>• Maximum user processes &gt; 1000000</li></ul> |
| GPU check           | gpu-check          | GPU and the v2 training engine are used. | GPUs are detected.                                                                                                                                                                                |

## Effect of a Fault Tolerance Check

- If the fault tolerance check is passed, the logs of the check items will be recorded, indicating that the check items are successful. You can search for the keyword **item** in the log file. A fault tolerance check minimizes reported runtime faults.
- If a fault tolerance check fails, check failure logs will be recorded. You can search for the keyword **item** in the log file to view the failure information.

If the number of job restarts does not reach the specified time, the job will be automatically issued again. You can search for keywords **error**, **exiting** to obtain the logs recording a restarted job that ends with a failure.

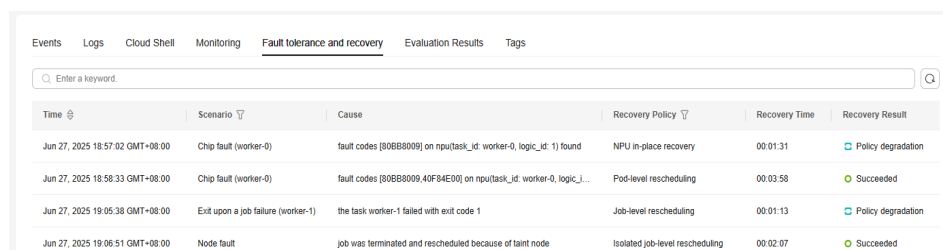
## Using reload ckpt to Resume an Interrupted Training

With fault tolerance enabled, if a training job is restarted due to a hardware fault, you can obtain the pre-trained model in the code to restore the training to the state before the restart. To do so, add reload ckpt to the code. For details, see [Resumable Training](#).

## Checking Fault Tolerance and Recovery Details

When a training job fault occurs (such as in-place NPU recovery and job-level rescheduling), the **Fault Recovery Details** tab appears on the job details page, recording the start and stop details of the training job.

Once you enable auto restart when creating a training job, you can check the training job restart details in the **Fault tolerance and recovery** tab of the training job details page.

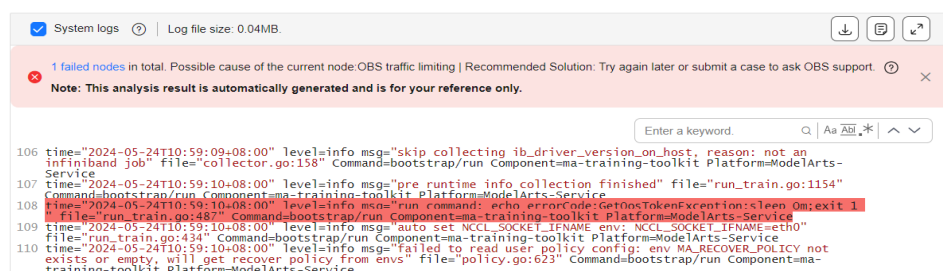
**Figure 7-27** Fault tolerance and recovery

| Time                            | Scenario                           | Cause                                                              | Recovery Policy                 | Recovery Time | Recovery Result    |
|---------------------------------|------------------------------------|--------------------------------------------------------------------|---------------------------------|---------------|--------------------|
| Jun 27, 2025 18:57:02 GMT+08:00 | Chip fault (worker-0)              | fault codes [808B8009] on nputask_id: worker-0, logic_id: 1) found | NPU in-place recovery           | 00:01:31      | Policy degradation |
| Jun 27, 2025 18:58:33 GMT+08:00 | Chip fault (worker-0)              | fault codes [808B8009,40F84E00] on nputask_id: worker-0, logic_... | Pod-level rescheduling          | 00:03:58      | Succeeded          |
| Jun 27, 2025 19:05:38 GMT+08:00 | Exit upon a job failure (worker-1) | the task worker-1 failed with exit code 1                          | Job-level rescheduling          | 00:01:13      | Policy degradation |
| Jun 27, 2025 19:06:51 GMT+08:00 | Node fault                         | job was terminated and rescheduled because of tainted node         | Isolated job-level rescheduling | 00:02:07      | Succeeded          |

## 7.11.2 Training Log Failure Analysis

If you encounter an issue during the execution of a ModelArts Standard training job, view logs first. In most scenarios, you can locate the issue based on the error information reported in logs.

If a training job fails, ModelArts Standard automatically identifies the failure cause and displays a message on the log page. The message consists of possible causes, recommended solutions, and error logs (marked in red).

**Figure 7-28** Identifying training faults

ModelArts Standard provides possible causes (for reference only) and solutions for some common training faults. Not all faults can be identified. For a distributed job, only the analysis result of the current node is displayed. To obtain the failure cause of a training job, check the analysis results of all nodes used by the training job.

To rectify common training faults, perform the following steps:

1. Rectify the fault based on the analysis and suggestions provided on the log page.
  - Solution 1: A troubleshooting document is provided for you to follow.
  - Solution 2: Copy the training job and run it again.
2. If the fault persists, analyze the error information in the logs to locate and rectify the fault.
3. If the provided solutions cannot rectify your fault, you can submit a service ticket for technical support.

## 7.11.3 Detecting Training Job Suspension

### Overview

A training job may be suspended due to unknown reasons. If the suspension cannot be detected promptly, resources cannot be released, leading to a waste. To minimize resource cost and improve user experience, ModelArts provides suspension detection for training jobs. With this function, suspension can be automatically detected and displayed on the log details page. You can also enable notification so that you can be promptly notified of job suspension.

### Detection Rules

Suspension detection determines whether a job is suspended based on the monitored job process status and resource usage. A coroutine is started to periodically monitor the changes of the two metrics. There are two types of suspension detection rules: single-instance detection and all-instance detection. Both apply simultaneously.

- **Single-instance detection**
  - Process status: If the process I/O of a single instance of a training job changes, the next detection period starts. If the process I/O remains unchanged in multiple detection periods, the resource usage detection starts.
  - Resource usage: If the process I/O of a single instance of a training job remains unchanged, the system collects the GPU or NPU usage within a certain period of time and determines whether the resource usage changes based on the variance and median of the GPU or NPU usage within the period. If the GPU usage is not changed, the job is suspended.
- **All-instance detection**

Resource usage: The system suspends a job if its GPU or NPU usage remains unchanged for a while and each instance's CPU usage stays below one core.

The system has the environment variable **MA\_HANG\_DETECT\_TIME** set to **30**. This means the job suspends if the system detects a metric issue for 30 minutes. To adjust this, update the value of the **MA\_HANG\_DETECT\_TIME** variable. For details, see [Managing Environment Variables of a Training Container](#).

---

#### CAUTION

- Due to the limitation of detection rules, there is a certain error probability in suspension detection. If the suspension is caused by the logic of job code (for example, long-time sleep), ignore it.
- 

### Constraints

Suspension can be detected only for training jobs that run on GPUs or NPUs.

### Procedure

Suspension detection is automatically performed during job running. No additional configuration is required. After detecting that a job is suspended, the system

displays a message on the training job details page, indicating that the job may be suspended. If you want to be notified of suspension (by SMS or email), enable event notification on the job creation page.

## Cases

### 1. Data replication suspension

#### Symptoms

The system stopped responding when **mox.file.copy\_parallel** was called to copy data.

#### Solution

- Run the following commands to copy files or folders:

```
import moxing as mox
mox.file.set_auth(is_secure=False)
```

- Run the following command to copy a single file that is greater than 5 GB:

```
from moxing.framework.file import file_io
```

Run **file\_io.LARGE\_FILE\_METHOD** to check the version of the MoXing API. Output value **1** indicates V1 and **2** indicates V2.

Run **file\_io.NUMBER\_OF\_PROCESSES=1** to resolve the issue for the V1 API.

To resolve the issue for the V2 API, use **file\_io.LARGE\_FILE\_METHOD = 1** to switch to V1 and perform operations required in V1. Alternatively, use **file\_io.LARGE\_FILE\_TASK\_NUM=1** to resolve this issue.

- Run the following command to copy a folder:

```
mox.file.copy_parallel(threads=0,is_processing=False)
```

### 2. Suspension before training

If a job was trained on multiple nodes and suspension occurred before the job started, add **os.environ["NCCL\_DEBUG"] = "INFO"** to the code to view the NCCL debugging information.

- Symptom 1**

The job was suspended before the NCCL debugging information was printed in logs.

#### Solution 1

Check the code for parameters such as **master\_ip** and **rank**. Ensure that these parameters are specified.

- Symptom 2**

According to the distributed training logs, some nodes contain GDR information, but some nodes do not. The suspension may be caused by GDR.

```
Logs of node A
modelarts-job-a7305e27-d1cf-4c71-ae6e-a12da6761d5a-worker-1:1136:1191 [2] NCCL INFO
Channel 00 : 3[5f000] -> 10[5b000] [receive] via NET/IB/0/GDRDMA
modelarts-job-a7305e27-d1cf-4c71-ae6e-a12da6761d5a-worker-1:1140:1196 [6] NCCL INFO
Channel 00 : 14[e1000] -> 15[e9000] via P2P/IPC
modelarts-job-a7305e27-d1cf-4c71-ae6e-a12da6761d5a-worker-1:1141:1187 [7] NCCL INFO
Channel 00 : 15[e9000] -> 11[5f000] via P2P/IPC
modelarts-job-a7305e27-d1cf-4c71-ae6e-a12da6761d5a-worker-1:1138:1189 [4] NCCL INFO
Channel 00 : 12[b5000] -> 14[e1000] via P2P/IPC
modelarts-job-a7305e27-d1cf-4c71-ae6e-a12da6761d5a-worker-1:1137:1197 [3] NCCL INFO
Channel 00 : 11[5f000] -> 16[2d000] [send] via NET/IB/0/GDRDMA
```

```
Logs of node B
modelarts-job-a7305e27-d1cf-4c71-ae6e-a12da6761d5a-worker-2:1139:1198 [2] NCCL INFO
Channel 00 : 18[5b000] -> 19[5f000] via P2P/IPC
modelarts-job-a7305e27-d1cf-4c71-ae6e-a12da6761d5a-worker-2:1144:1200 [7] NCCL INFO
Channel 00 : 23[e9000] -> 20[b5000] via P2P/IPC
modelarts-job-a7305e27-d1cf-4c71-ae6e-a12da6761d5a-worker-2:1142:1196 [5] NCCL INFO
Channel 00 : 21[be000] -> 17[32000] via P2P/IPC
modelarts-job-a7305e27-d1cf-4c71-ae6e-a12da6761d5a-worker-2:1143:1194 [6] NCCL INFO
Channel 00 : 22[e1000] -> 21[be000] via P2P/IPC
modelarts-job-a7305e27-d1cf-4c71-ae6e-a12da6761d5a-worker-2:1141:1191 [4] NCCL INFO
Channel 00 : 20[b5000] -> 22[e1000] via P2P/IPC
```

### Solution 2

Set `os.environ["NCCL_NET_GDR_LEVEL"] = '0'` at the beginning of the program to disable GDR or ask the O&M personnel to add the GDR information to the affected nodes.

#### – Symptom 3

Communication information such as "Got completion with error 12, opcode 1, len 32478, vendor err 129" was displayed. The current network was unstable.

### Solution 3

Add the following environment variables:

- **NCCL\_IB\_GID\_INDEX=3**: enables RoCEv2. RoCEv1 is enabled by default. However, RoCEv1 does not support congestion control on switches, which may lead to packet loss. In addition, later-version switches do not support RoCEv1, leading to a RoCEv1 failure.
- **NCCL\_IB\_TC=128**: enables data packets to be transmitted through the queue 4 of switches, which is RoCE-compliant.
- **NCCL\_IB\_TIMEOUT=22**: enables a longer timeout interval. Generally, there is a network interruption lasting about 5s if the network is unstable and then the timeout message is returned. Change the timeout interval to 22s, indicating that the timeout message will be returned in about 20s ( $4.096 \mu\text{s} \times 2^{\text{timeout}}$ ).

### 3. Suspension during training

#### – Symptom 1

According to the logs of the nodes on which a training job ran, an error occurred on a node but the job did not exit, leading to the job suspension.

### Solution 1

Check the error cause and rectify the fault.

#### – Symptom 2

The job was stuck in sync-batch-norm or the training speed was lowered down. If sync-batch-norm is enabled for PyTorch, the training speed is lowered down because all node data must be synchronized on each batch normalization layer in every iteration, which leads to heavy communication traffic.

### Solution 2

Disable sync-batch-norm, or upgrade the PyTorch version to 1.10.

- **Symptom 3**

The job is stuck in TensorBoard, and the following error is reported:  
writer = Summarywriter('./path)/to/log')

**Solution 3**

Set a local path for storage, for example, **cache/tensorboard**. Do not store data in OBS.

- **Symptom 4**

When PyTorch DataLoader is used to read data, the job is stuck in data reading, and logs stop to update.

**Solution 4**

When the DataLoader is used to read data, reduce the value of **num\_worker**.

#### 4. Suspension in the Last Training Epoch

**Symptoms**

Logs showed that an error occurred in split data. As a result, processes are in different epochs, and uncompleted processes are suspended because they do not receive response from other processes. As shown in the following figure, some processes are in epoch 48 while others are in epoch 49 at the same time.

```
loss exit lane:0.12314446270465851
step loss is 0.29470521211624146
[2022-04-26 13:57:20,757][INFO][train_epoch]:Rank:2 Epoch:[48][20384/all] Data Time 0.000(0.000)
Net Time 0.705(0.890) Loss 0.3403(0.3792) LR 0.00021887
[2022-04-26 13:57:20,757][INFO][train_epoch]:Rank:1 Epoch:[48][20384/all] Data Time 0.000(0.000)
Net Time 0.705(0.891) Loss 0.3028(0.3466) LR 0.00021887
[2022-04-26 13:57:20,757][INFO][train_epoch]:Rank:4 Epoch:[49][20384/all] Data Time 0.000(0.147)
Net Time 0.705(0.709) Loss 0.3364(0.3414) LR 0.00021887
[2022-04-26 13:57:20,758][INFO][train_epoch]:Rank:3 Epoch:[49][20384/all] Data Time 0.000 (0.115)
Net Time 0.706(0.814) Loss 0.3345(0.3418) LR 0.00021887
[2022-04-26 13:57:20,758][INFO][train_epoch]:Rank:0 Epoch:[49][20384/all] Data Time 0.000(0.006)
Net Time 0.704(0.885) Loss 0.2947(0.3566) LR 0.00021887
[2022-04-26 13:57:20,758][INFO][train_epoch]:Rank:7 Epoch:[49][20384/all] Data Time 0.001 (0.000)
Net Time 0.706 (0.891) Loss 0.3782(0.3614) LR 0.00021887
[2022-04-26 13:57:20,759][INFO][train_epoch]:Rank:5 Epoch:[48][20384/all] Data Time 0.000(0.000)
Net Time 0.706(0.891) Loss 0.5471(0.3642) LR 0.00021887
[2022-04-26 13:57:20,763][INFO][train_epoch]:Rank:6 Epoch:[49][20384/all] Data Time 0.000(0.000)
Net Time 0.704(0.891) Loss 0.2643(0.3390) LR 0.00021887
stage 1 loss 0.4600560665130615 mul_cls_loss loss:0.01245919056236744 mul_offset_loss
0.44759687781333923 origin stage2_loss 0.048592399805784225
stage 1 loss:0.4600560665130615 stage 2 loss:0.048592399805784225 loss exit
lane:0.10233864188194275
```

**Solution**

Split tensors to align data.

## 7.11.4 Training Job Restart Upon Suspension

If a stable training job runs properly for a while and then gets suspended without a hardware fault, you can restart it to fix the issue. However, the container cannot be automatically stopped during suspension. In this case, auto restart cannot be used and you must set the job to restart upon suspension. When configured this way, ModelArts forcibly stops the user process in the container upon detecting suspension, then runs the training job boot command again after stopping the process. Resource scheduling is not involved during this restart; the job restarts in the original container.

For details about the suspension detection rules, see [Detecting Training Job Suspension](#).

To avoid losing training progress, ensure your code can resume training from where it is interrupted, and then enable unconditional auto restart to optimize compute usage. For details, see [Resumable Training](#).

If auto restart is triggered during training, the system records the restart information. You can check the fault recovery details on the training job details page. For details, see [Training Job Fault Tolerance Check](#).

## Notes and Constraints

To prevent compute waste from invalid restarts, training jobs can restart upon suspension a maximum of three consecutive times.

## Enabling Restart Upon Suspension

To enable restart upon suspension, use either the console or API.

- Using the console

On the **Create Training Job** page, enable auto restart and restart upon suspension. After enabling restart upon suspension, the system restarts the training job in the container when a suspension is detected. This process does not involve resource scheduling, so **Restarts** is not affected.

**Figure 7-29** Enabling restart upon suspension

### Fault tolerance and recovery

☒ Auto Restart

If a training job encounters an exception due to envi  
your code logic supports resumable training before i

Maximum number of restarts ⓘ

— | 3 | +

☐ Unconditional auto restart

As long as a training exception is detected, Mor

☒ Restart Upon Suspension

ModelArts continuously monitors job processes  
may result in false reports. By enabling this feat

- Using an API

When creating a training job through an API, input the **fault-tolerance/job-retry-num** and **fault-tolerance/hang-retry** fields in **annotations** of the **metadata** field. To enable auto restart and set the number of restarts, set



**fault-tolerance/job-retry-num** to an integer between 1 and 128. For restart upon suspension, set **fault-tolerance/hang-retry** to **true**.

**Table 7-40** Parameters

| Parameter   | Mandatory | Type               | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-------------|-----------|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| kind        | Yes       | String             | <b>Description:</b> Type of a training job.<br><b>Constraints:</b> N/A.<br><b>Options:</b> <ul style="list-style-type: none"><li>• <b>job</b>: common job</li><li>• <b>edge_job</b>: edge job</li><li>• <b>hetero_job</b>: heterogeneous job</li><li>• <b>mrs_job</b>: MRS job</li><li>• <b>autosearch_job</b>: auto search job</li><li>• <b>diag_job</b>: diagnosis job</li><li>• <b>visualization_job</b>: visualization job</li></ul> <b>Default value:</b> job                                                                                                                                                                                                                                                            |
| annotations | No        | Map<String,String> | <b>Description:</b> Advanced functions of a training job.<br><b>Constraints:</b> The options are as follows: <ul style="list-style-type: none"><li>• <b>job_template</b>: Template RL (heterogeneous job)</li><li>• <b>fault-tolerance/job-retry-num</b>: 3 (number of retries upon a fault)</li><li>• <b>fault-tolerance/job-unconditional-retry</b>: true (unconditional restart)</li><li>• <b>fault-tolerance/hang-retry</b>: true (restart upon suspension)</li><li>• <b>jupyter-lab/enable</b>: true (JupyterLab training application)</li><li>• <b>tensorboard/enable</b>: true (TensorBoard training application)</li><li>• <b>mindstudio-insight/enable</b>: true (MindStudio Insight training application)</li></ul> |

```
{
 "kind": "job",
 "metadata": {
 "annotations": {
 "fault-tolerance/job-retry-num": "3",
 "fault-tolerance/hang-retry": "true"
 }
 }
}
```

```
}
}
```

## 7.11.5 Resumable Training

### Overview

Resumable training indicates that an interrupted training job can be automatically resumed from the checkpoint where the previous training was interrupted. This method is applicable to model training that takes a long time.

The checkpoint mechanism enables resumable training.

During model training, training results (including but not limited to epochs, model weights, optimizer status, and scheduler status) are continuously saved. In this way, an interrupted training job can be automatically resumed from the checkpoint where the previous training was interrupted.

To resume a training job, load a checkpoint and use the checkpoint information to initialize the training status. To do so, add `reload ckpt` to the code.

### Implementing Resumable Training in ModelArts Standard

To resume model training or incrementally train a model in ModelArts Standard, configure training output.

When creating a training job, set the data path to the training output, save checkpoints in this data path, and set **Predownload** to **Yes**. If you set **Predownload** to **Yes**, the system automatically downloads the **checkpoint** file in the training output data path to a local directory of the training container before the training job is started.

**Figure 7-30** Configuring training output

Training Configuration

Input ⓘ

| Parameter Name                      | Input Source                          | Obtained from     | Operation                |
|-------------------------------------|---------------------------------------|-------------------|--------------------------|
| <input type="text" value="Please"/> | <input type="text" value="Please"/> ⓘ | Hyperparameters ▾ | <a href="#">Deletion</a> |

[Add](#)

Output ⓘ

| Parameter Name                      | Output Path                           | Obtained from     | Predownload to Container... ⓘ | Operation                |
|-------------------------------------|---------------------------------------|-------------------|-------------------------------|--------------------------|
| <input type="text" value="Please"/> | <input type="text" value="Please"/> ⓘ | Hyperparameters ▾ | No ▾                          | <a href="#">Deletion</a> |

Enable fault tolerance check (auto restart) for resumable training. On the training job creation page, enable **Auto Restart**. If the environment pre-check fails, the hardware is not functional, or the training job fails, ModelArts will automatically issue the training job again.

### reload ckpt for PyTorch

- Use either of the following methods to save a PyTorch model.
  - Save model parameters only.

```
state_dict = model.state_dict()
torch.save(state_dict, path)
```

- Save the entire model (not recommended).  
`torch.save(model, path)`

- Save the data generated during model training at regular intervals based on steps and time.

The data includes the network weight, optimizer weight, and epoch, which will be used to resume the interrupted training.

```
checkpoint = {
 "net": model.state_dict(),
 "optimizer": optimizer.state_dict(),
 "epoch": epoch
}
if not os.path.isdir('model_save_dir'):
 os.makedirs('model_save_dir')
torch.save(checkpoint, 'model_save_dir/ckpt_{}.pth'.format(str(epoch)))
```

- Check the complete code example below.

```
import os
import argparse
parser = argparse.ArgumentParser()
parser.add_argument("--train_url", type=str)
args, unparsed = parser.parse_known_args()
args = parser.parse_known_args()
train_url is set to /home/ma-user/modelarts/outputs/train_url_0.
train_url = args.train_url

Check whether there is a model file in the output path. If there is no file, the model will be trained
from the beginning by default. If there is a model file, the CKPT file with the maximum epoch value
will be loaded as the pre-trained model.
if os.listdir(train_url):
 print('> load last ckpt and continue training!!')
 last_ckpt = sorted([file for file in os.listdir(train_url) if file.endswith(".pth")])[-1]
 local_ckpt_file = os.path.join(train_url, last_ckpt)
 print('last_ckpt:', last_ckpt)
 # Load the checkpoint.
 checkpoint = torch.load(local_ckpt_file)
 # Load the parameters that can be learned by the model.
 model.load_state_dict(checkpoint['net'])
 # Load optimizer parameters.
 optimizer.load_state_dict(checkpoint['optimizer'])
 # Obtain the saved epoch. The model will continue to be trained based on the epoch value.
 start_epoch = checkpoint['epoch']
start = datetime.now()
total_step = len(train_loader)
for epoch in range(start_epoch + 1, args.epochs):
 for i, (images, labels) in enumerate(train_loader):
 images = images.cuda(non_blocking=True)
 labels = labels.cuda(non_blocking=True)
 # Forward pass
 outputs = model(images)
 loss = criterion(outputs, labels)
 # Backward and optimize
 optimizer.zero_grad()
 loss.backward()
 optimizer.step()
 ...

 # Save the network weight, optimizer weight, and epoch during model training.
 checkpoint = {
 "net": model.state_dict(),
 "optimizer": optimizer.state_dict(),
 "epoch": epoch
 }
 if not os.path.isdir(train_url):
 os.makedirs(train_url)
 torch.save(checkpoint, os.path.join(train_url, 'ckpt_best_{}.pth'.format(epoch)))
```

## 7.11.6 Enabling Unconditional Auto Restart

To prevent training failures and delays, use unconditional auto restart. This feature automatically restarts a failed job, regardless of the cause, improving training success rates and job stability. To prevent invalid restarts, the system limits unconditional restarts to three consecutive attempts.

To avoid losing training progress, ensure your code can resume training from where it is interrupted, and then enable unconditional auto restart to optimize compute usage. For details, see [Resumable Training](#).

If auto restart is triggered during training, the system records the restart information. You can check the fault recovery details on the training job details page. For details, see [Training Job Fault Tolerance Check](#).

### Procedure

You can enable unconditional auto restart either on the console or through an API.

- Using the console  
On the training job creation page, enable **Auto Restart** and select **Unconditional auto restart**. If **Unconditional auto restart** is enabled, the training job will be restarted unconditionally once the system detects a training exception. If you enable auto restart but do not select **Unconditional auto restart**, the training job will only automatically restart if it encounters environmental issues. In case of any other problems, the status of the training job will become **Failed**.

**Figure 7-31** Enabling unconditional auto restart

#### Fault tolerance and recovery

☒ Auto Restart  
If a training job encounters an exception due to your code logic supports resumable training by

Maximum number of restarts ?  
- | 3 | +

☒ Unconditional auto restart  
As long as a training exception is detected

- Using an API  
When creating a training job through an API, input the **fault-tolerance/job-retry-num** and **fault-tolerance/job-unconditional-retry** fields in **annotations** of the **metadata** field. To enable auto restart and set the number of restarts, set **fault-tolerance/job-retry-num** to an integer between 1 and 128. For unconditional auto restart, set **fault-tolerance/job-unconditional-retry** to **true**.

**Table 7-41** Parameters

| Parameter   | Mandatory | Type               | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-------------|-----------|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| kind        | Yes       | String             | <p><b>Description:</b> Type of a training job.</p> <p><b>Constraints:</b> N/A.</p> <p><b>Options:</b></p> <ul style="list-style-type: none"> <li>• <b>job:</b> common job</li> <li>• <b>edge_job:</b> edge job</li> <li>• <b>hetero_job:</b> heterogeneous job</li> <li>• <b>mrs_job:</b> MRS job</li> <li>• <b>autosearch_job:</b> auto search job</li> <li>• <b>diag_job:</b> diagnosis job</li> <li>• <b>visualization_job:</b> visualization job</li> </ul> <p><b>Default value:</b> job</p>                                                                                                                                                                                                                                                 |
| annotations | No        | Map<String,String> | <p><b>Description:</b> Advanced functions of a training job.</p> <p><b>Constraints:</b> The options are as follows:</p> <ul style="list-style-type: none"> <li>• <b>job_template:</b> Template RL (heterogeneous job)</li> <li>• <b>fault-tolerance/job-retry-num:</b> 3 (number of retries upon a fault)</li> <li>• <b>fault-tolerance/job-unconditional-retry:</b> true (unconditional restart)</li> <li>• <b>fault-tolerance/hang-retry:</b> true (restart upon suspension)</li> <li>• <b>jupyter-lab/enable:</b> true (JupyterLab training application)</li> <li>• <b>tensorboard/enable:</b> true (TensorBoard training application)</li> <li>• <b>mindstudio-insight/enable:</b> true (MindStudio Insight training application)</li> </ul> |

```
{
 "kind": "job",
 "metadata": {
 "annotations": {
 "fault-tolerance/job-retry-num": "8",
 "fault-tolerance/job-unconditional-retry": "true"
 }
 }
}
```

## 7.12 Configuring Supernode Affinity Group Instances

### Description

In foundation model training, multiple parallel strategies boost efficiency and scalability using distributed computing. Model parallelism uses AllReduce communication, while MoE expert parallelism uses all-to-all communication. Both require high network bandwidth between processing units (PUs). These communication steps often become bottlenecks due to hardware limitations, limiting training performance.

Snt9b23 uses the HCCS bus to connect NPUs across multiple compute nodes, creating supernodes. Within each supernode, a fully interconnected network known as a superplane enhances AI task communication. With supernode hardware, models can adjust parallel strategies like model parallelism or MoE expert parallelism to speed up training with higher bandwidth.

ModelArts provides a **supernode affinity feature** that acts as a scheduling policy. This feature organizes AI training jobs to align with the hardware network of compute resources, maximizing the high bandwidth and low latency of supernodes for better training efficiency. Algorithm engineers can effortlessly use supernode hardware through simple setups.

### Principles

ModelArts uses affinity groups to schedule training jobs on supernodes and maximize their ultra-high bandwidth. An affinity group is a group of instances that you want to connect through a hyperplane in a training job. Instances in the same affinity group run on the same supernode.

To boost training efficiency, place PUs that need high bandwidth in the same affinity group.

For Tensor Parallelism (TP), in older hardware, TP is often set to 8, linking eight PUs per node with HCCS. In supernodes, more PUs connect via HCCS, allowing for larger TP domains.

Prerequisite: In distributed training, each instance must use all PUs on the node.

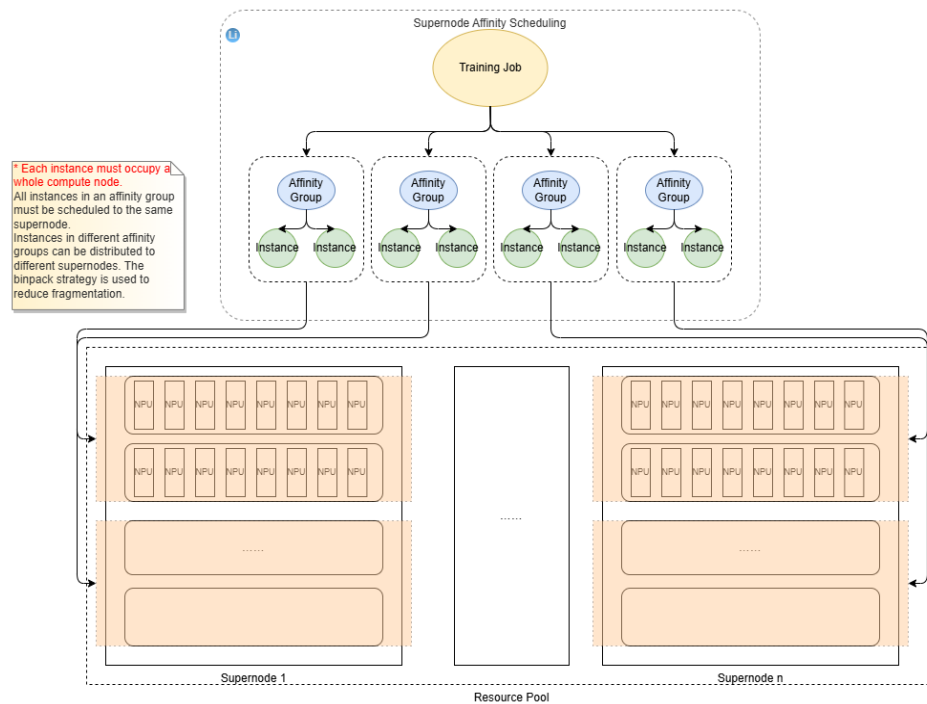
If TP is 32, then 32 PUs form one TP domain, and each node has 16 PUs. Here, two instances must share the same supernode, forming an affinity group.

You only need to set the new parameter **Supernode Affinity Group Instances** to **2** when creating a training job.

The training job runs only if the resource pool's available supernodes meet these conditions:

1. Available nodes  $\geq$  Job instances
2. Allocatable affinity groups  $\geq$  Required affinity groups

**Figure 7-32** Supernode affinity scheduling for a training job



If the training job needs eight instances, with **Supernode Affinity Group Instances** set to 2. Here are the available resources:

- Resource pool A  
Supernode 1: maximum nodes: 8; available nodes: 6  
Supernode 2: maximum nodes: 8; available nodes: 2
- Resource pool B  
Supernode 3: maximum nodes: 8; available nodes: 7  
Supernode 4: maximum nodes: 8; available nodes: 1

Both pools have eight available nodes, meeting the **first** condition. However, supernode 3 in pool B can allocate three affinity groups, while supernode 4 can allocate none. The **second** condition is not met. Thus, only pool A can run the training job. If the job goes to pool B, it will wait in the queue due to insufficient resources.

## Constraints

- The affinity group feature only works with dedicated resource pools that have Ascend Snt9b23 supernodes.
- In distributed training, each instance must use all PUs on the node.

## Configuration Method

When you **create a training job** on the **ModelArts console**, selecting a dedicated resource pool and an Snt9b23 flavor allows you to set **Supernode Affinity Group Instances** during **Configuring a Dedicated Resource Pool**.

Notes for setting **Supernode Affinity Group Instances**:

- The default value is **1**, and the maximum value cannot exceed the maximum number of nodes on a single supernode.
- You must set the number of instances as an integral multiple of **Supernode Affinity Group Instances**. Otherwise, the training job cannot be created.
- After choosing a dedicated resource pool, you can check its supernode specifications, like total nodes, total PUs, available nodes, and available PUs. This helps you decide on **Supernode Affinity Group Instances**.

## 7.13 Managing Model Training Jobs

### 7.13.1 Viewing Training Job Details

1. Log in to the [ModelArts console](#).
2. In the navigation pane, choose **Model Training > Training Jobs**.  
In the job list, click **Export** to export training job details in a certain time range as an Excel file. A maximum of 200 rows of data can be exported.  
In the search box above the training job list, filter jobs by attributes like status, mode, type, or priority.
3. In the training job list, click the target job name to switch to the training job details page.
4. On the left of the training job details page, view basic job settings and algorithm parameters.
  - **Basic job settings**

**Table 7-42** Basic job settings

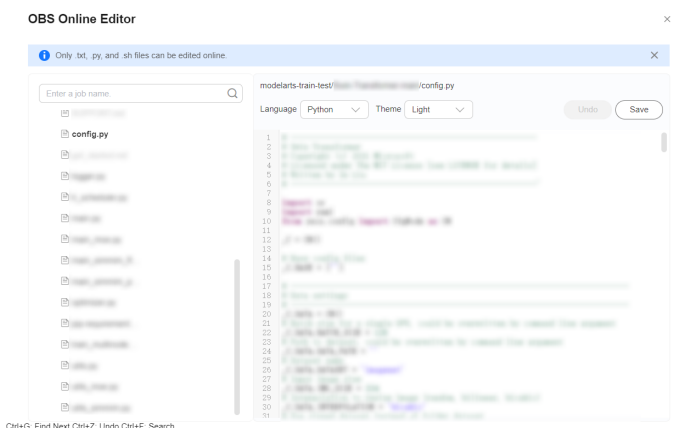
| Parameter    | Description                                                                                                                                                                                                                                              |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Job ID       | Unique ID of the training job.                                                                                                                                                                                                                           |
| Status       | Status of the training job.                                                                                                                                                                                                                              |
| Created      | Time when the training job is created.                                                                                                                                                                                                                   |
| Duration     | Duration of a training job, which is the total duration of Kubernetes resources in the entire lifecycle of a training job.                                                                                                                               |
| Retries      | Number of times that the training job automatically restarts upon a fault. This parameter is only available when <b>Auto Restart</b> is enabled during training job creation.<br><b>Number of restarts/Maximum number of restarts</b> is displayed here. |
| Description  | Description of the training job.<br>You can click the edit icon to update the description of a training job.                                                                                                                                             |
| Job Priority | Priority of the training job.                                                                                                                                                                                                                            |



| Parameter  | Description                                                                                          |
|------------|------------------------------------------------------------------------------------------------------|
| Preemption | This parameter is only displayed when <b>Preemption</b> is enabled when you create the training job. |

– **Algorithm parameters**

**Table 7-43** Algorithm parameters

| Parameter            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Algorithm Name       | Algorithm used in the training job. You can click the algorithm name to go to the algorithm details page.                                                                                                                                                                                                                                                                                                                                                                                                     |
| Preset images        | Preset image used by the training job. This parameter is available only for training jobs created using a preset image.                                                                                                                                                                                                                                                                                                                                                                                       |
| Custom image         | Custom image used by the training job. This parameter is available only for training jobs created using a custom image.                                                                                                                                                                                                                                                                                                                                                                                       |
| Code Directory       | <p>OBS path to the code directory of the training job. You can click <b>Edit Code</b> on the right to edit the training script code in <b>OBS Online Editor</b>. <b>OBS Online Editor</b> is not available for a training job in the <b>Pending</b>, <b>Creating</b>, or <b>Running</b> status.</p>  <p><b>NOTE</b><br/>This parameter is not supported when you use a subscribed algorithm to create a training job.</p> |
| Boot File            | <p>Location where the training boot file is stored.</p> <p><b>NOTE</b><br/>This parameter is not supported when you use a subscribed algorithm to create a training job.</p>                                                                                                                                                                                                                                                                                                                                  |
| User ID              | ID of the user who runs the container.                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| Local Code Directory | Path to the training code in the training container.                                                                                                                                                                                                                                                                                                                                                                                                                                                          |

| Parameter                                     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-----------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Work Directory                                | Path to the training boot file in the training container.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| Compute Nodes                                 | Number of instances for the training job.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| Dedicated resource pool                       | Dedicated resource pool information. This parameter is available only when a training job uses a dedicated resource pool.                                                                                                                                                                                                                                                                                                                                                                                                                          |
| Compute Node ID                               | Names and IP addresses of the compute nodes used by the training job. This parameter is only displayed when the training job uses a dedicated resource pool.                                                                                                                                                                                                                                                                                                                                                                                       |
| Specifications                                | <p>Instance specifications used by the training job. This parameter is only displayed when the training job does not use custom specifications of a dedicated resource pool.</p> <p>This parameter shows the instance specifications for the training job, both allocated to the training containers and chosen during job creation. The actual resources used are usually less than those chosen at job creation. This happens because the job's internal containers use some resources. These containers help run the training job smoothly.</p> |
| Customized Specifications                     | <p>Instance specifications used by the training job. This parameter is only displayed when the training job uses custom specifications of a dedicated resource pool.</p> <p>This parameter shows the custom resource and instance specifications selected for the training job.</p>                                                                                                                                                                                                                                                                |
| Input > Input Path                            | OBS path where the input data is stored.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Input > Parameter Name                        | Input path parameter specified in the algorithm code.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Input > Obtained from                         | Method of obtaining the training job input.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| Input > Local Path (Training Parameter Value) | Path for storing the input data in the ModelArts backend container. After the training is started, ModelArts downloads the data stored in OBS to the backend container.                                                                                                                                                                                                                                                                                                                                                                            |
| Output > Output Path                          | OBS path where the output data is stored.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |

| Parameter                                      | Description                                                          |
|------------------------------------------------|----------------------------------------------------------------------|
| Output > Parameter Name                        | Output path parameter specified in the algorithm code.               |
| Output > Obtained from                         | Method of obtaining the training job output.                         |
| Output > Local Path (Training Parameter Value) | Path for storing the output data in the ModelArts backend container. |
| Hyperparameter                                 | Hyperparameters used in the training job.                            |
| Environment Variable                           | Environment variables for the training job.                          |

5. On the training details page, manage event notifications of the training job.

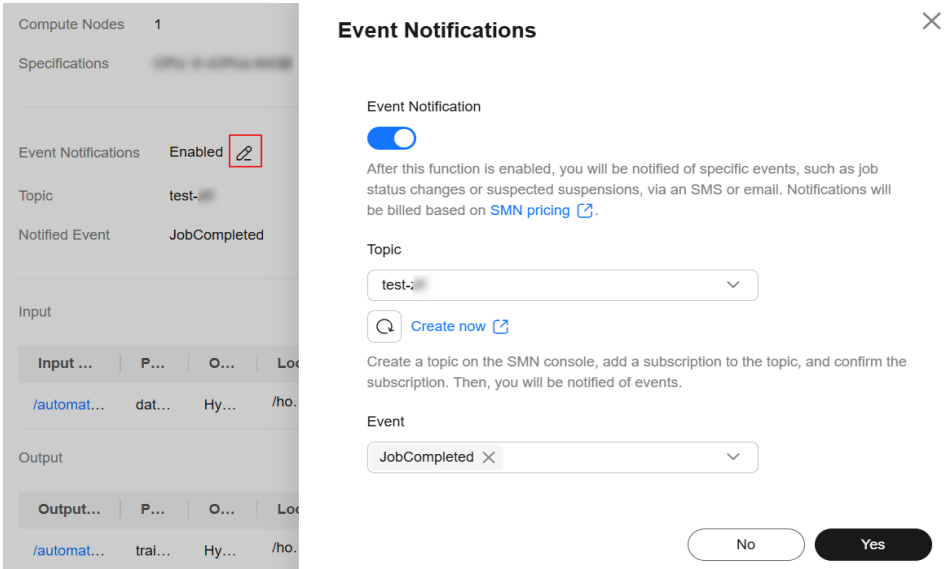
 **NOTE**

- Event notifications cannot be configured for training jobs in the **Completed**, **Failed**, **Abnormal**, or **Terminated** state.
- To set up event notifications, you need permission to view jobs.
- Only the updated training status is notified for modification events.

After event notification is enabled, you will be notified of a specific event, such as a job status change or suspected suspension, through an SMS message or email. Notifications will be billed based on SMN pricing. For details, see [Billing](#).

- If event notification has been enabled for a training job, you can click  next to **Enabled** to modify or disable event notification.

Figure 7-33 Modifying event notification



- If event notification has not been enabled for a training job, you can click  next to **Disabled** to enable event notification.

Figure 7-34 Configuring event notification

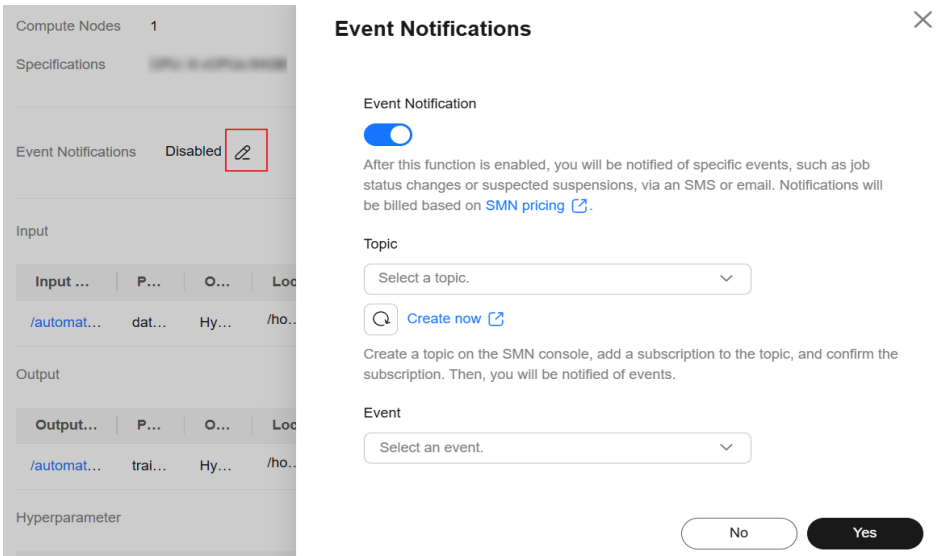


Table 7-44 Event notification parameters

| Parameter | Description                                                                                                                                                                                                                                                                                                                                      |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Topic     | Topic name of event notification. You can select a topic from the drop-down list or click <b>Create now</b> to create a topic on the SMN console.<br><b>NOTE</b><br>You can create a topic on the SMN console, add a subscription to it, and confirm the subscription status. Once these steps are completed, you will be notified of the event. |

| Parameter | Description                                                                                                                                                                                                                                           |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Event     | Select events you want to subscribe to. Examples: <b>JobStarted</b> , <b>JobCompleted</b> , <b>JobFailed</b> , <b>JobTerminated</b> , and <b>JobHanged</b> .<br><b>NOTE</b><br>Only training jobs using GPUs or NPUs support <b>JobHanged</b> events. |

## 7.13.2 Visualizing the Training Job Process

### Description

When training machine learning models, you must track the status of your training jobs. Traditional methods often fail to show all necessary details, making it hard to follow progress accurately. With this feature, you can check the real-time status of a training job across multiple dimensions like job scheduling, environment setup, and execution progress. The visual interface allows you to check the entire lifecycle of a training job, helping you monitor progress, adjust parameters, and improve both efficiency and user experience.

### Constraints

The main phases of the job process remain constant, while the subphases can vary. For example, if a training job lacks input data, the "input is downloading" sub-phase under "Environment Preparing" will be absent. Similarly, if the "job initializing environment is checking" event is missing, its corresponding sub-phase within the "Job Running" phase will also be omitted.

### Checking Training Phases

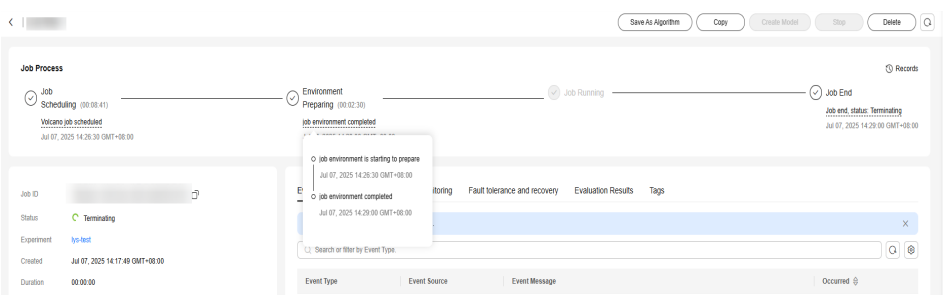
On the top of the training job details page, the training job module of ModelArts Standard displays the job process details. There are four main phases: **Job Scheduling**, **Environment Preparing**, **Job Running**, and **Job End**.

You can check the subphases of each main phase.

- **Job Scheduling** logs all job activities, including successful or failed creations, scheduled jobs, and their exact timestamps.
- **Environment Preparing** documents essential steps for preparing the environment. It includes starting the setup, checking initial conditions, downloading training code, and completing the process, along with timestamps.
- **Job Running** tracks job details, including start and end times for each training step.
- **Job End** captures essential details like the date and time of the job's outcome.

Click **Records** in the upper right corner of the job process to see all its activity logs. This allows you to track details like repeated preemptions, rescheduling, or restarts during the job's execution.

Figure 7-35 Job process



### 7.13.3 Viewing the Resource Usage of a Training Job

#### Description

In the **Monitoring** tab of the training job details page, you can view the CPU, GPU, and NPU usage for the job or a single node.

#### Notes and Constraints

You can view the monitoring data of the entire training period. For details, see [Table 7-45](#).

Usage data of training job resources is stored for 30 days before being automatically deleted.

#### Operations

1. On the [ModelArts console](#), choose **Model Training > Training Jobs** from the navigation pane.
2. In the training job list, click the name of the target job to go to the training job details page.
3. Click the **Monitoring** tab to view the resource usage of the training job. You can monitor resources in the following dimensions. [Table 7-45](#) describes the metrics.
  - Job monitoring: Monitors the overall resource usage of the current training job. You can select the last 15 minutes, last 30 minutes, last 1 hour, last 6 hours, last 12 hours, or last 24 hours, or specify a period.
  - Task monitoring: Monitors the resource usage of the training job's nodes. You can select the last 15 minutes, last 30 minutes, last 1 hour, last 6 hours, last 12 hours, or last 24 hours, or specify a period.

**Table 7-45** Training job monitoring metrics

| Category | Metric                | Parameter                           | Description                                                                                                                                                                                                                                                       | Unit | Value Range | Presentation Form |
|----------|-----------------------|-------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|-------------|-------------------|
| CPU      | CPU Usage             | ma_container_cpu_util               | CPU usage of a measured object                                                                                                                                                                                                                                    | %    | 0%–100%     | Line chart        |
|          | Used cores            | ma_container_cpu_used_core          | Number of CPU cores used by a measured object                                                                                                                                                                                                                     | Core | ≥ 0         | Bar chart         |
| Memory   | Physical Memory Usage | ma_container_memory_util            | Percentage of the used physical memory to the total physical memory                                                                                                                                                                                               | %    | 0%–100%     | Line chart        |
|          | Used Physical Memory  | ma_container_memory_used_meg        | Physical memory that has been used by a measured object ( <b>container_memory_working_set_bytes</b> in the current working set)<br>(Memory usage in a working set = Active anonymous page and cache, and file-backed page ≤ <b>container_memory_usage_bytes</b> ) | MB   | ≥ 0         | Bar chart         |
| GPU      | GPU Usage             | ma_container_gpu_util               | GPU usage of a measured object                                                                                                                                                                                                                                    | %    | 0%–100%     | Line chart        |
|          | GPU Memory Usage      | ma_container_gpu_mem_util           | Percentage of the used GPU memory to the total GPU memory                                                                                                                                                                                                         | %    | 0%–100%     | Line chart        |
|          | Used GPU Memory       | ma_container_gpu_mem_used_megabytes | GPU memory used by a measured object                                                                                                                                                                                                                              | MB   | ≥ 0         | Bar chart         |

| Category | Metric                | Parameter                              | Description                                                                                                                                                                                 | Unit    | Value Range | Presentation Form |
|----------|-----------------------|----------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|-------------|-------------------|
| NPU      | NPU Usage             | ma_container_npu_ai_core_util          | AI core usage of Ascend AI processors                                                                                                                                                       | %       | 0%–100%     | Line chart        |
|          | NPU Memory Usage      | ma_container_npu_memory_util           | Percentage of the used NPU memory to the total NPU memory (To be replaced by <b>ma_container_npu_ddr_memory_util</b> for Snt3 series, and <b>ma_container_npu_hbm_util</b> for Snt9 series) | %       | 0%–100%     | Line chart        |
|          | Used NPU Memory       | ma_container_npu_memory_used_megabytes | NPU memory used by a measured object (To be replaced by <b>ma_container_npu_ddr_memory_usage_bytes</b> for Snt3 series, and <b>ma_container_npu_hbm_usage_bytes</b> for Snt9 series)        | MB      | ≥ 0         | Bar chart         |
| Network  | Network Downlink Rate | ma_container_network_receive_bytes     | Inbound traffic rate of a measured object                                                                                                                                                   | Bytes/s | ≥ 0         | Line chart        |
|          | Network Uplink Rate   | ma_container_network_transmit_bytes    | Outbound traffic rate of a measured object                                                                                                                                                  | Bytes/s | ≥ 0         | Line chart        |
| Disk     | Disk Read Rate        | ma_container_disk_read_kilobytes       | Volume of data read from a disk per second                                                                                                                                                  | KB/s    | ≥ 0         | Line chart        |



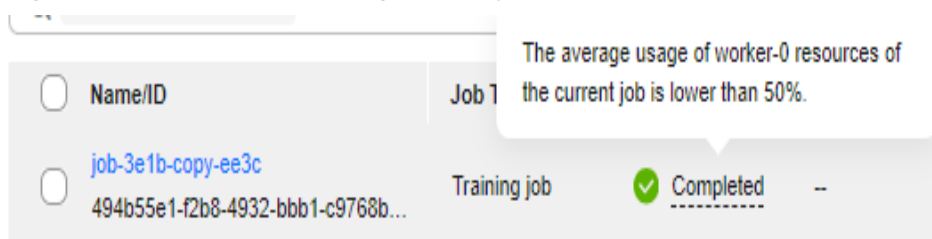
| Category | Metric          | Parameter                         | Description                                   | Unit | Value Range | Presentation Form |
|----------|-----------------|-----------------------------------|-----------------------------------------------|------|-------------|-------------------|
|          | Disk Write Rate | ma_container_disk_write_kilobytes | Volume of data written into a disk per second | KB/s | $\geq 0$    | Line chart        |

For more metrics, see [Viewing Monitoring Metrics of a Training Job](#).

## Alarms of Job Resource Usage

You can view the job resource usage on the training job list page. If the average GPU/NPU usage of the job's worker-0 instance is lower than 50%, an alarm is displayed in the training job list.

**Figure 7-36** Job resource usage in the job list



The job resource usage here involves only GPU and NPU resources. The method of calculating the average GPU/NPU usage of a job's worker-0 instance is: Summarize the usage of each GPU/NPU accelerator card at each time point of the job's worker-0 instance and calculate the average value.

## Improving Job Resource Utilization

- Increasing the value of **batch\_size** increases GPU and NPU usage. You must decide the batch size that will not cause a memory overflow.
- If the time for reading data in a batch is longer than the time for GPUs or NPUs to calculate data in a batch, GPU or NPU usage may fluctuate. In this case, optimize the performance of data reading and data augmentation. For example, read data in parallel or use tools such as NVIDIA Data Loading Library (DALI) to improve the data augmentation speed.
- If a model is large and frequently saved, GPU or NPU usage is affected. In this case, do not save models frequently. Similarly, make sure that other non-GPU/NPU operations, such as log printing and training metric saving, do not affect the training process for too much time.

## 7.13.4 Viewing the Model Evaluation Result

After a training job has been executed, ModelArts evaluates your model and provides optimization diagnosis and suggestions.

- When you use a built-in algorithm to create a training job, you can view the evaluation result without any configurations. The system automatically provides optimization suggestions based on your model metrics. Read the suggestions and guidance on the page carefully to further optimize your model.
- For a training job created by writing a training script or using a custom image, you need to add the evaluation code to the training code so that you can view the evaluation result and diagnosis suggestions after the training job is complete.

### NOTE

- Only validation sets of the image type are supported.
- You can add the evaluation code only when the training scripts of the following frequently-used frameworks are used:
  - TF-1.13.1-python3.6
  - TF-2.1.0-python3.6
  - PyTorch-1.4.0-python3.6

This section describes how to use the evaluation code in a training job. To adapt and modify the training code, three steps are involved, [Adding the Output Path](#), [Copying the Dataset to the Local Host](#), and [Mapping the Dataset Path to OBS](#).

### Adding the Output Path

The code for adding the output path is simple. That is, add a path for storing the evaluation result file to the code, which is called **train\_url**, that is, the training output path on the console. Add **train\_url** to the analysis function and use **save\_path** to obtain **train\_url**. The sample code is as follows:

```
FLAGS = tf.app.flags.FLAGS
tf.app.flags.DEFINE_string('model_url', '', 'path to saved model')
tf.app.flags.DEFINE_string('data_url', '', 'path to output files')
tf.app.flags.DEFINE_string('train_url', '', 'path to output files')
tf.app.flags.DEFINE_string('adv_param_json',
 '{"attack_method":"FGSM","eps":40}',
 'params for adversarial attacks')
FLAGS(sys.argv, known_only=True)

...

analyse
res = analyse(
 task_type=task_type,
 pred_list=pred_list,
 label_list=label_list,
 name_list=file_name_list,
 label_map_dict=label_dict,
 save_path=FLAGS.train_url)
```

## Copying the Dataset to the Local Host

Copying a dataset to the local host is to prevent the OBS connection from being interrupted due to long-time access. Therefore, copy the dataset to the local host before performing operations.

There are two methods for copying datasets. The recommended method is to use the OBS path.

- OBS path (recommended)  
Call the `copy_parallel` API of MoXing to copy the corresponding OBS path.
- Dataset in ModelArts data management (manifest file format)  
Call the `copy_manifest` API of MoXing to copy the file to the local host and obtain the path of the new manifest file. Then, use SDK to parse the new manifest file.

### NOTE

ModelArts data management is being upgraded and is invisible to users who have not used data management. It is recommended that new users store their training data in OBS buckets.

```
if data_path.startswith('obs://'):
 if 'manifest' in data_path:
 new_manifest_path, _ = mox.file.copy_manifest(data_path, '/cache/data/')
 data_path = new_manifest_path
 else:
 mox.file.copy_parallel(data_path, '/cache/data/')
 data_path = '/cache/data/'
print('----- download dataset success -----')
```

## Mapping the Dataset Path to OBS

The actual path of the image file, that is, the OBS path, needs to be entered in the JSON body. Therefore, after analysis and evaluation are performed on the local host, the original local dataset path needs to be mapped to the OBS path, and the new list needs to be sent to the analysis API.

If the OBS path is used as the input of **data\_url**, you only need to replace the string of the local path.

```
if FLAGS.data_url.startswith('obs://'):
 for idx, item in enumerate(file_name_list):
 file_name_list[idx] = item.replace(data_path, FLAGS.data_url)
```

If the manifest file is used, the original manifest file needs to be parsed again to obtain the list and then the list is sent to the analysis API.

```
if or FLAGS.data_url.startswith('obs://'):
 if 'manifest' in FLAGS.data_url:
 file_name_list = []
 manifest, _ = get_sample_list(
 manifest_path=FLAGS.data_url, task_type='image_classification')
 for item in manifest:
 if len(item[1]) != 0:
 file_name_list.append(item[0])
```

An example code for image classification that can be used to create training jobs is as follows:

```
import json
import logging
```

```
import os
import sys
import tempfile

import h5py
import numpy as np
from PIL import Image

import moxing as mox
import tensorflow as tf
from deep_moxing.framework.manifest_api.manifest_api import get_sample_list
from deep_moxing.model_analysis.api import analyse, tmp_save
from deep_moxing.model_analysis.common.constant import TMP_FILE_NAME

logging.basicConfig(level=logging.DEBUG)

FLAGS = tf.app.flags.FLAGS
tf.app.flags.DEFINE_string('model_url', '', 'path to saved model')
tf.app.flags.DEFINE_string('data_url', '', 'path to output files')
tf.app.flags.DEFINE_string('train_url', '', 'path to output files')
tf.app.flags.DEFINE_string('adv_param_json',
 '{"attack_method": "FGSM", "eps": 40}',
 'params for adversarial attacks')
FLAGS(sys.argv, known_only=True)

def _preprocess(data_path):
 img = Image.open(data_path)
 img = img.convert('RGB')
 img = np.asarray(img, dtype=np.float32)
 img = img[np.newaxis, :, :, :]
 return img

def softmax(x):
 x = np.array(x)
 orig_shape = x.shape
 if len(x.shape) > 1:
 # Matrix
 x = np.apply_along_axis(lambda x: np.exp(x - np.max(x)), 1, x)
 denominator = np.apply_along_axis(lambda x: 1.0 / np.sum(x), 1, x)
 if len(denominator.shape) == 1:
 denominator = denominator.reshape((denominator.shape[0], 1))
 x = x * denominator
 else:
 # Vector
 x_max = np.max(x)
 x = x - x_max
 numerator = np.exp(x)
 denominator = 1.0 / np.sum(numerator)
 x = numerator.dot(denominator)
 assert x.shape == orig_shape
 return x

def get_dataset(data_path, label_map_dict):
 label_list = []
 img_name_list = []
 if 'manifest' in data_path:
 manifest, _ = get_sample_list(
 manifest_path=data_path, task_type='image_classification')
 for item in manifest:
 if len(item[1]) != 0:
 label_list.append(label_map_dict.get(item[1][0]))
 img_name_list.append(item[0])
 else:
 continue
 else:
 label_name_list = os.listdir(data_path)
```

```
label_dict = {}
for idx, item in enumerate(label_name_list):
 label_dict[str(idx)] = item
 sub_img_list = os.listdir(os.path.join(data_path, item))
 img_name_list += [
 os.path.join(data_path, item, img_name) for img_name in sub_img_list
]
 label_list += [label_map_dict.get(item)] * len(sub_img_list)
return img_name_list, label_list

def deal_ckpt_and_data_with_obs():
 pb_dir = FLAGS.model_url
 data_path = FLAGS.data_url

 if pb_dir.startswith('obs://'):
 mox.file.copy_parallel(pb_dir, '/cache/ckpt/')
 pb_dir = '/cache/ckpt'
 print('----- download success -----')
 if data_path.startswith('obs://'):
 if '.manifest' in data_path:
 new_manifest_path, _ = mox.file.copy_manifest(data_path, '/cache/data/')
 data_path = new_manifest_path
 else:
 mox.file.copy_parallel(data_path, '/cache/data/')
 data_path = '/cache/data/'
 print('----- download dataset success -----')
 assert os.path.isdir(pb_dir), 'Error, pb_dir must be a directory'
 return pb_dir, data_path

def evalution():
 pb_dir, data_path = deal_ckpt_and_data_with_obs()
 index_file = os.path.join(pb_dir, 'index')
 try:
 label_file = h5py.File(index_file, 'r')
 label_array = label_file['labels_list'][:].tolist()
 label_array = [item.decode('utf-8') for item in label_array]
 except Exception as e:
 logging.warning(e)
 logging.warning('index file is not a h5 file, try json.')
 with open(index_file, 'r') as load_f:
 label_file = json.load(load_f)
 label_array = label_file['labels_list'][:].
 label_map_dict = {}
 label_dict = {}
 for idx, item in enumerate(label_array):
 label_map_dict[item] = idx
 label_dict[idx] = item
 print(label_map_dict)
 print(label_dict)

 data_file_list, label_list = get_dataset(data_path, label_map_dict)

 assert len(label_list) > 0, 'missing valid data'
 assert None not in label_list, 'dataset and model not match'

 pred_list = []
 file_name_list = []
 img_list = []

 for img_path in data_file_list:
 img = _preprocess(img_path)
 img_list.append(img)
 file_name_list.append(img_path)

 config = tf.ConfigProto()
 config.gpu_options.allow_growth = True
 config.gpu_options.visible_device_list = '0'
```

```
with tf.Session(graph=tf.Graph(), config=config) as sess:
 meta_graph_def = tf.saved_model.loader.load(
 sess, [tf.saved_model.tag_constants.SERVING], pb_dir)
 signature = meta_graph_def.signature_def
 signature_key = 'predict_object'
 input_key = 'images'
 output_key = 'logits'
 x_tensor_name = signature[signature_key].inputs[input_key].name
 y_tensor_name = signature[signature_key].outputs[output_key].name
 x = sess.graph.get_tensor_by_name(x_tensor_name)
 y = sess.graph.get_tensor_by_name(y_tensor_name)
 for img in img_list:
 pred_output = sess.run([y], {x: img})
 pred_output = softmax(pred_output[0])
 pred_list.append(pred_output[0].tolist())

label_dict = json.dumps(label_dict)
task_type = 'image_classification'

if FLAGS.data_url.startswith('obs://'):
 if 'manifest' in FLAGS.data_url:
 file_name_list = []
 manifest, _ = get_sample_list(
 manifest_path=FLAGS.data_url, task_type='image_classification')
 for item in manifest:
 if len(item[1]) != 0:
 file_name_list.append(item[0])
 for idx, item in enumerate(file_name_list):
 file_name_list[idx] = item.replace(data_path, FLAGS.data_url)
 # analyse
 res = analyse(
 task_type=task_type,
 pred_list=pred_list,
 label_list=label_list,
 name_list=file_name_list,
 label_map_dict=label_dict,
 save_path=FLAGS.train_url)

if __name__ == "__main__":
 evaluation()
```

## 7.13.5 Viewing Training Job Events

Any key event of a training job will be recorded at the backend after the training job is displayed for you. You can check events on the training job details page.

This helps you better understand the running process of a training job and locate faults more accurately when a task exception occurs. The following job events are supported:

- Training job created.
- Training job failures:
- Preparations timed out. The possible cause is that the cross-region algorithm synchronization or creating shared storage timed out.
- The training job is queuing and awaiting resource allocation.
- Failed to be queued.
- The training job starts to run.
- Training job executed.
- Failed to run the training job.
- The training job is preempted.

- The system detects that your training job may be suspended. Go to the job details page to view the cause and handle the issue.
- The training job has been restarted.
- The training job has been manually stopped.
- The training job has been stopped. (Maximum running duration: x hours)
- The training job has been manually deleted.
- Billing information synchronized.
- [worker-0] The training environment is being pre-checked.
- [worker-0] [Duration: second] Pre-check completed.
- [worker-0] [Duration: second] Pre-check failed. Error: xxx
- [worker-0] [Duration: second] Pre-check failed. Error: xxx
- [worker-0] The training code is being downloaded.
- [worker-0] [Duration: second] Training code downloaded.
- [worker-0] [Duration: second] Failed to download the training code. Failure cause:
- [worker-0] The training input is being downloaded.
- [worker-0] [Duration: second] Training input (parameter: xxx) downloaded.
- [worker-0] [Duration: second] Failed to download the training input (parameter: xxx). Failure cause:
- [worker-0] Python dependency packages are being installed. Import the following files:
- [worker-0] [Duration: second] Python dependency packages installed. Import the following files:
- [worker-0] The training job starts to run.
- [worker-0] Training job executed.
- [worker-0] The training input is being uploaded.
- [worker-0] [Duration: second] Training output (parameter: xxx) uploaded.

During the training process, key events can be manually or automatically refreshed.

## Notes and Constraints

The system automatically stores training job events for 30 days, and any expired events will be deleted.

## Procedure

1. On the [ModelArts console](#), choose **Model Training** > **Training Jobs** from the navigation pane.
2. In the training job list, click the name of the target job to go to the training job details page.
3. Click **Events** to view events.

Figure 7-37 Events

Events

Search or filter by Event Type.

| Event Type |    | Event Message                                                    | Occurred          |
|------------|----|------------------------------------------------------------------|-------------------|
| Normal     | -- | Training job completed.                                          | Nov 27, 2024 0... |
| Normal     | -- | [Job: job] ExecuteAction: Start to execute action CompleteJob    | Nov 27, 2024 0... |
| Normal     | -- | [worker-0][time used: 42.833s] Upload training output(paramet... | Nov 27, 2024 0... |

7.13.6 Viewing Training Job Logs

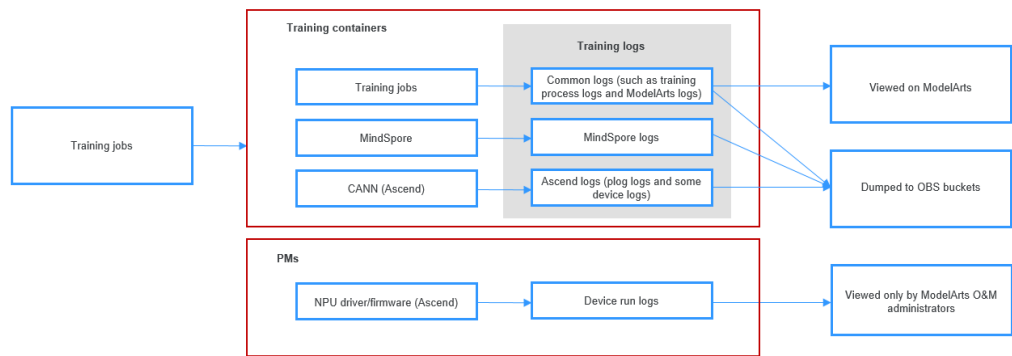
Overview

Training logs record the runtime process and exception information of training jobs and provide useful details for fault location. The standard output and standard error information in your code are displayed in training logs. If you encounter an issue during the execution of a ModelArts training job, view logs first. In most scenarios, you can locate the issue based on the error information reported in logs.

Training logs include common training logs and Ascend logs.

- Common Logs:** When resources other than Ascend are used for training, only common training logs are generated. Common logs include the logs for **pip-requirement.txt**, training process, and ModelArts.
- Ascend Logs:** When Ascend resources are used for training, device logs, plog logs, proc log for single-card training logs, MindSpore logs, and common logs are generated.

Figure 7-38 ModelArts training logs



NOTE

Separate MindSpore logs are generated only in the MindSpore+Ascend training scenario. Logs of other AI engines are contained in common logs.

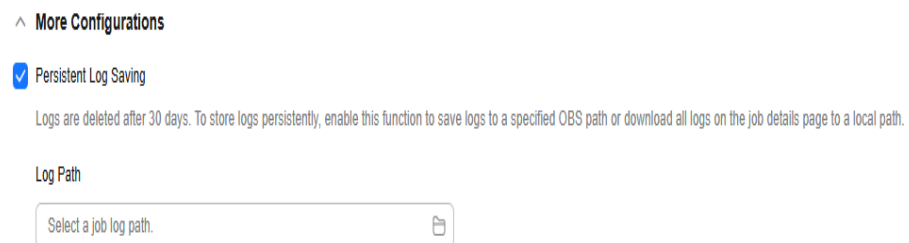


## Retention Period

Logs are classified into the following types based on the retention period:

- Real-time logs: generated during training job running and can be viewed on the ModelArts training job details page.
- Historical logs: After a training job is completed, you can view its historical logs on the ModelArts training job details page. ModelArts automatically stores the logs for 30 days.
- Permanent logs: These logs are dumped to your OBS bucket. When creating a training job, you can enable persistent log saving and set a job log path for dumping.

**Figure 7-39** Enabling persistent log saving



Real-time logs and historical logs have no difference in content. In the Ascend training scenario, permanent logs contain Ascend logs, which are not displayed on ModelArts.

## Common Logs

Common logs include the logs for **pip-requirement.txt**, training process, and ModelArts Standard.

**Table 7-46** Common log types

| Type                                             | Description                                                                                             |
|--------------------------------------------------|---------------------------------------------------------------------------------------------------------|
| Training process log                             | Standard output of your training code.                                                                  |
| Installation logs for <b>pip-requirement.txt</b> | If <b>pip-requirement.txt</b> is defined in training code, pip package installation logs are generated. |
| ModelArts logs                                   | ModelArts logs are used by O&M personnel to locate service faults.                                      |

The format of a common log file is as follows. **task id** is the node ID of a training job.

Unified log format: modelarts-job-[job id]-[task id].log

Example: log/modelarts-job-95f661bd-1527-41b8-971c-eca55e513254-worker-0.log

- Single-node training jobs generate a log file, and **task id** defaults to **worker-0**.

- Distributed training generates multiple node log files, which are distinguished by **task id**, such as **worker-0** and **worker-1**.

Common logs include the logs for **pip-requirement.txt**, training process, and ModelArts.

ModelArts logs can be filtered in the common log file **modelarts-job-[job id]-[task id].log** using the following keywords: **[ModelArts Service Log]** or **Platform=ModelArts-Service**.

- Type 1: [ModelArts Service Log] xxx  
[ModelArts Service Log][init] download code\_url: s3://dgg-test-user/snt9-test-cases/mindspore/lenet/
- Type 2: time="xxx" level="xxx" msg="xxx" file="xxx" Command=xxx  
Component=xxx Platform=xxx  
time="2021-07-26T19:24:11+08:00" level=info msg="start the periodic upload task, upload period = 5 seconds " file="upload.go:46" Command=obs/upload Component=ma-training-toolkit  
Platform=ModelArts-Service

## Ascend Logs

Ascend logs are generated when Ascend resources are used to for training. When Ascend resources are used for training, device logs, plog logs, proc logs for single-card training logs, MindSpore logs, and common logs are generated.

Common logs in the Ascend training scenario include the logs for **pip-requirement.txt**, **ma-pre-start**, **davincirun**, training process, and ModelArts.

The following is an example of the Ascend log structure:

```
obs://dgg-test-user/snt9-test-cases/log-out/ # Job log path
├── modelarts-job-9ccf15f2-6610-42f9-ab99-059ba049a41e
│ ├── ascend
│ │ ├── process_log
│ │ │ ├── rank_0
│ │ │ └── plog # Plog logs
│ │ └── device-0 ... # Device logs
│ └── mindspore ... # MindSpore logs
├── modelarts-job-95f661bd-1527-41b8-971c-eca55e513254-worker-0.log # Common logs
└── modelarts-job-95f661bd-1527-41b8-971c-eca55e513254-proc-rank-0-device-0.txt # proc log for
single-card training logs
```

**Table 7-47** Ascend log description

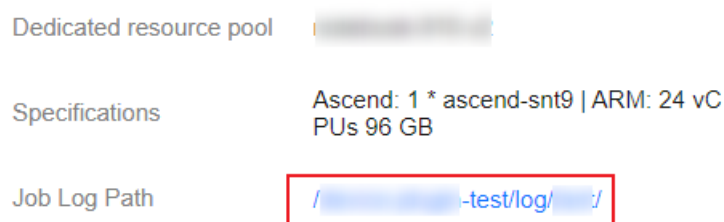
| Type        | Description                                                                                                                                                                                                                                                                                                                                                                                                                        | Name                                                                                                                                                                                                                   |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Device logs | <p>User process AICPU and HCCP logs generated on the device and sent back to the host (training container).</p> <p>If any of the following situations occur, device logs cannot be obtained:</p> <ul style="list-style-type: none"> <li>• The compute node restarts unexpectedly.</li> <li>• The compute node stops expectedly.</li> </ul> <p>After the training process ends, the log is generated in the training container.</p> | <p><b>~/ascend/log/device-{device-id}/device-{pid}_{timestamp}.log</b></p> <p>In the preceding command, <b>pid</b> indicates the user process ID on the host.</p> <p>Example:<br/>device-166_20220718191853764.log</p> |
| Plog logs   | <p>User process logs, for example, ACL/GE.</p> <p>Plog logs are generated in the training container.</p>                                                                                                                                                                                                                                                                                                                           | <p><b>~/ascend/log/plog/plog-{pid}_{timestamp}.log</b></p> <p>In the preceding command, <b>pid</b> indicates the user process ID on the host.</p> <p>Example:<br/>plog-166_20220718191843620.log</p>                   |

| Type           | Description                                                                                                                                                                                                                                                                                                                                                                    | Name                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| proc log       | <p><b>proc log</b> is a redirection file of single-node training logs, helping you quickly obtain logs of a compute node. <b>proc log</b> for training using a preset MindSpore image and ranktable is generated in the training container and automatically saved in OBS. Training jobs using custom MindSpore images or other frameworks do not involve <b>proc log</b>.</p> | <p><b>[modelarts-job-uuid]-proc-rank-[rank id]-device-[device logic id].txt</b></p> <ul style="list-style-type: none"> <li>• <b>device id</b> indicates the ID of the NPU used in the training job. The value is 0 for a single NPU and 0 to 7 for eight NPUs. For example, if the Ascend specification is 8*Snt9, the value of device id ranges from 0 to 7. If the Ascend specification is 1*Snt9, the value of device id is 0.</li> <li>• <b>rank id</b> indicates the global NPU ID of the training job. The value ranges from 0 to the number of instances multiplied by the number of NPUs minus 1. If a single instance is used, the value of <b>rank id</b> is the same as that of <b>device id</b>.</li> </ul> <p>Example:<br/>modelarts-<br/>job-95f661bd-1527-41b8-971c-<br/>eca55e513254-proc-rank-0-<br/>device-0.txt</p> |
| MindSpore logs | <p>Separate MindSpore logs are generated in the MindSpore +Ascend training scenario. MindSpore logs are generated in the training container.</p>                                                                                                                                                                                                                               | <p>For details about MindSpore logs, visit the <a href="#">MindSpore official website</a>.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

| Type                 | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Name                                                                                                                                                                                                                                                                                                                                                                                                                          |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Common training logs | <p>Common training logs are generated in the <b>/home/ma-user/modelarts/log</b> directory of the training container and automatically uploaded to OBS. The common training logs include these types:</p> <ul style="list-style-type: none"><li>• Logs for <b>ma-pre-start</b> (specific to Ascend training): If the <b>ma-pre-start</b> script is defined, the script execution log is generated.</li><li>• Logs for <b>davincirun</b> (specific to Ascend training): log generated when the Ascend training process is started using the <b>davincirun.py</b> file</li><li>• Training process logs: standard output of user training code</li><li>• Logs for <b>pip-requirement.txt</b>: If <b>pip-requirement.txt</b> is defined in training code, pip package installation logs are generated.</li><li>• ModelArts logs: used by O&amp;M personnel to locate service faults.</li></ul> | <p>Contained in the <b>modelarts-job-[job id]-[task id].log</b> file.</p> <p><b>task id</b> indicates the instance ID. If a single node is used, the value is <b>worker-0</b>. If multiple nodes are used, the value is <b>worker-0</b>, <b>worker-1</b>, ..., or <b>worker-{n-1}</b>. <b>n</b> indicates the number of instances.</p> <p>Example:</p> <p>modelarts-job-95f661bd-1527-41b8-971c-eca55e513254-worker-0.log</p> |

In the Ascend training scenario, after the training process exits, ModelArts uploads the log files in the training container to the OBS directory specified by **Job Log Path**. On the job details page, you can obtain the job log path and click the OBS address to go to the OBS console to check logs.

**Figure 7-40** Job Log Path



You can run the **ma-pre-start** script to modify the default environment variable configurations.

```
ASCEND_GLOBAL_LOG_LEVEL=3 # Log level, 0 for debug, 1 for info, 2 for warning, and 3 for error.
ASCEND_SLOG_PRINT_TO_STDOUT=1 # Whether to display plog logs. The value 1 indicates that plog logs
 # are displayed by default.
ASCEND_GLOBAL_EVENT_ENABLE=1 # Event log level, 0 for disabling event logging and 1 for enabling
 # event logging.
```

Place the **ma-pre-start.sh** or **ma-pre-start.py** script in the directory at the same level as the training boot file.

Before the training boot file is executed, the system executes the **ma-pre-start** script in **/home/work/user-job-dir/**. This method can be used to update the Ascend RUN package installed in the container image or set some additional global environment variables required for training.

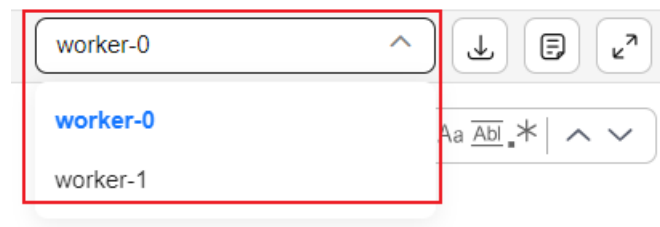
## Viewing Training Job Logs

On the training job details page, you can preview logs, download logs, search for logs by keyword, and filter system logs in the log pane.

- **Previewing logs**

You can preview training logs on the system log pane. If multiple compute nodes are used, you can choose the target node from the drop-down list on the right.

**Figure 7-41** Viewing logs of different compute nodes



If a log file is oversized, the system displays only the latest logs in the log pane. To view all logs, click the link in the upper part of the log pane, which will direct you to a new page. Then you will be redirected to a new page.

**Figure 7-42** Viewing all logs



### NOTE

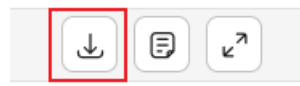
- If the total size of all logs exceeds 500 MB, the log page may be frozen. In this case, download the logs to view them locally.
- A log preview link can be accessed by anyone within one hour after it is generated. You can share the link with others.
- **Ensure that no privacy information is contained in the logs. Otherwise, information leakage may occur.**

- Downloading logs

Training logs are retained for only 30 days. To permanently store logs, click the download icon in the upper right corner of the log pane. You can download the logs of multiple compute nodes in a batch. You can also enable **Persistent Log Saving** and set a log path when you create a training job. In this way, the logs will be automatically stored in the specified OBS path.

If a training job is created on Ascend compute nodes, certain system logs cannot be downloaded in the training log pane. To obtain these logs, go to the **Job Log Path** you set when you created the training job.

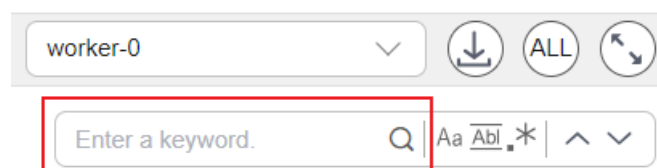
**Figure 7-43** Downloading logs



- Searching for logs by keyword

In the upper right corner of the log pane, enter a keyword in the search box to search for logs, as shown in [Figure 7-44](#).

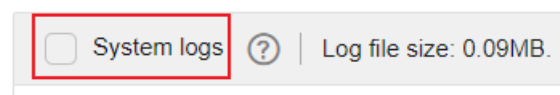
**Figure 7-44** Searching for logs by keyword



The system will highlight the keyword and redirect you between search results. Only the logs loaded in the log pane can be searched for. If the logs are not fully displayed (see the message displayed on the page), obtain all the logs by downloading them or clicking the full log link and then search for the logs. On the page redirected by the full log link, press **Ctrl+F** to search for logs.

- Filtering system logs

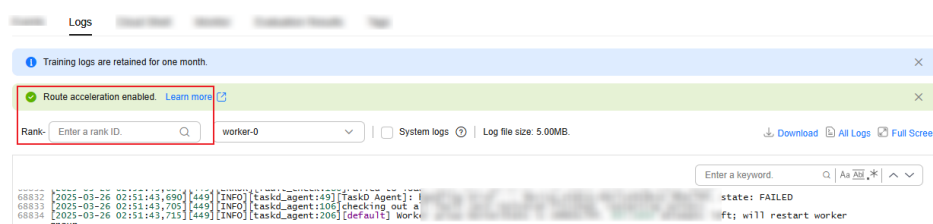
**Figure 7-45** System logs



If **System logs** is selected, system logs and user logs are displayed. If **System logs** is deselected, only user logs are displayed.

- Viewing dynamic route acceleration logs

If your training job has three or more instances, the **ROUTE\_PLAN** environment variable is set to **true**, and you use Ascend resources, you can view the dynamic route acceleration logs by rank ID and check if dynamic route acceleration is enabled.

**Figure 7-46** Viewing dynamic route acceleration logs

## 7.13.7 Priority of a Training Job

When using a dedicated resource pool for training jobs, you can set the job priority when creating a training job or adjust the priority when a job is in the **Pending** state for a long time. By adjusting the job priority, you can reduce the job queuing duration.

### Overview

Some training jobs, such as unimportant tests or experiments, are of low priority. In this case, you need to prioritize training tasks (jobs). Jobs with higher priority are scheduled before those with lower priority.

You can adjust the job execution sequence by configuring the priority of training jobs to ensure normal running of important services at peak hours.

### Constraints

- You can set the priority of a training job only if it is created using a dedicated resource pool.
- The job priority can be set to **1**, **2**, or **3**. A larger number indicates a higher priority. The default priority is **1**, and the highest priority is **3**. To set the priority to **3**, you must have the permission. For details about how to configure the permission, see [Assigning the Permission to Set the Highest Job Scheduling Priority to an IAM User](#).
- If you want to change the priority of a training job in the **Pending** state, it must be queued for resources in a dedicated resource pool.

### Configuring the Priority

On the **Create Training Job** page, select **Increase job scheduling priority**. The priority can be set to **1**, **2**, or **3**. A larger number indicates a higher priority. The default priority is **1**, and the highest priority is **3**.

### Changing the Priority

- On the training job list page, find the target training job in the **Pending** status.
- Click the target training job to access its details page.
- Click  next to **Job Priority**. In the displayed dialog box, change the priority and click **OK**.

ModelArts Standard manages training job queues using a priority system.



1. For jobs with equal priority, scheduling follows these rules:
  - If resources are enough, jobs run based on their submission order using the FIFO method.
  - If resources fall short, some jobs from separate logical pools might get prioritized instead, disrupting strict FIFO execution.
2. For jobs with adjusted priority, scheduling follows this rule:  
After a training job is delivered to a resource pool, its priority cannot be changed.

## Assigning the Permission to Set the Highest Job Scheduling Priority to an IAM User

You can set the job priority to **1** or **2** by default. Once permission to set the highest priority is granted, you can set it to **1**, **2**, or **3**. If you have been assigned with the **modelarts:trainJob:setHighPriority** permission, you can set this parameter to **3**.

1. Log in to the [Huawei Cloud console](#) as a tenant user, hover over your username in the upper right corner, and choose **Identity and Access Management** from the drop-down list to switch to the IAM management console.
2. On the IAM console, choose **Permissions > Policies/Roles** from the navigation pane, click **Create Custom Policy** in the upper right corner, and configure the following parameters.
  - **Policy Name:** Enter a custom policy name, for example, **Allowing Users to Set the Highest Job Priority**.
  - **Policy View:** Select **Visual editor**.
  - **Policy Content:** Select **Allow, ModelArts Service, modelarts:trainJob:setHighPriority**, and default resources.
3. In the navigation pane, choose **User Groups**. Then, click **Authorize** in the **Operation** column of the target user group. On the **Authorize User Group** page, select the custom policies created in [2](#), and click **Next**. Then, select the scope and click **OK**.

After the configuration, all users in the user group have the permission to set the training job priority to **1** to **3**.

If no user group is available, create a user group, add users using the user group management function, and configure authorization. If the target user is not in a user group, you can add the user to a user group through the user group management function.

## 7.13.8 Using Cloud Shell to Debug a Production Training Job

ModelArts Standard provides Cloud Shell, which allows you to log in to a running container to debug training jobs in the production environment.

### Constraints

Only dedicated resource pools allow logging in to training containers using Cloud Shell. The training job must be running.

## Preparation: Assigning the Cloud Shell Permission to an IAM User

1. Log in to the [Huawei Cloud console](#) as a tenant user, hover over your username in the upper right corner, and choose **Identity and Access Management** from the drop-down list to switch to the IAM management console.
2. On the IAM console, choose **Permissions > Policies/Roles** from the navigation pane, click **Create Custom Policy** in the upper right corner, and configure the following parameters.
  - **Policy Name:** Enter a custom policy name, for example, **Using Cloud Shell to access a running job**.
  - **Policy View:** Select **Visual editor**.
  - **Policy Content:** Select **Allow, ModelArts Service, modelarts:trainJob:exec**, and default resources.
3. In the navigation pane, choose **User Groups**. Then, click **Authorize** in the **Operation** column of the target user group. On the **Authorize User Group** page, select the custom policies created in 2, and click **Next**. Then, select the scope and click **OK**.

After the configuration, all users in the user group have the permission to use Cloud Shell to log in to a running training container.

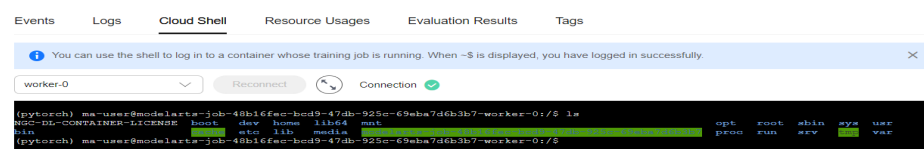
If no user group is available, create a user group, add users using the user group management function, and configure authorization. If the target user is not in a user group, you can add the user to a user group through the user group management function.

## Using Cloud Shell

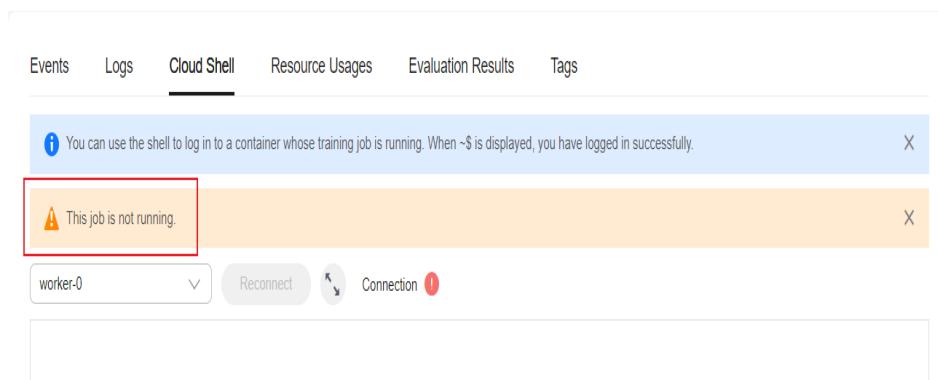
1. Configure parameters based on [Preparation: Assigning the Cloud Shell Permission to an IAM User](#).
2. Log in to the [ModelArts console](#). In the navigation pane, choose **Model Training > Training Jobs**.
3. In the training job list, click the name of the target job to go to the training job details page.
4. On the training job details page, click the **Cloud Shell** tab and log in to the training container.

Verify that the login is successful, as shown in the following figure.

Figure 7-47 Cloud Shell page



If the job is not running or the permission is insufficient, Cloud Shell cannot be used. In this case, locate the fault as prompted.

**Figure 7-48** Error message**NOTE**

If you encounter a path display issue when logging in to Cloud Shell, press **Enter** to resolve the problem.

**Figure 7-49** Path display issue

```
ind/model/1$ [97c6-b87f-4410-9f74-18a8b1d0ff9d-59x451kz-6548f94565-1rjgs:/home/mi
```

## Keeping a Training Job Running

You can only log in to Cloud Shell when the training job is in **Running** state. This section describes how to log in to a running training container through Cloud Shell.

### Using the sleep Command

- For training jobs using a preset image

When creating a training job, set **Algorithm Type** to **Custom algorithm** and **Boot Mode** to **Preset image**, add **sleep.py** to the code directory, and use the script as the boot file. The training job keeps running for 60 minutes. You can access the container through Cloud Shell for debugging.

#### Example of **sleep.py**

```
import os
os.system('sleep 60m')
```

**Figure 7-50** Using a preset image

The screenshot shows the configuration interface for a training job. The 'Algorithm Type' is set to 'Custom algorithm'. The 'Boot Mode' is set to 'Preset image'. The 'Code Directory' is '/modelarts-train/'. The 'Boot File' is '/modelarts-train/sleep.py', which is highlighted with a red box. The 'Local Code Directory' is '/home/ma-user/'. The 'Work Directory' is also highlighted with a red box.

★ Algorithm Type: Custom algorithm, My algorithm, My subscription

★ Boot Mode: Preset image, Custom image

★ Code Directory: /modelarts-train/ Select

★ Boot File: /modelarts-train/sleep.py Select

Local Code Directory: /home/ma-user/

Work Directory: Select

- For training jobs using a custom image  
When creating a training job, set **Algorithm Type** to **Custom algorithm** and **Boot Mode** to **Custom image**, and enter **sleep 60m** in **Boot Command**. The training job keeps running for 60 minutes. You can access the container through Cloud Shell for debugging.

**Figure 7-51** Using a custom image

The screenshot shows the configuration interface for a training job. The 'Algorithm Type' is set to 'Custom algorithm'. The 'Boot Mode' is set to 'Custom image'. The 'Image' is selected. The 'Code Directory' is empty. The 'User ID' is '1000'. The 'Boot Command' is 'sleep 60m', which is highlighted with a red box.

★ Algorithm Type: Custom algorithm, My algorithm, My subscription

★ Boot Mode: Preset image, Custom image

★ Image: Select

Code Directory: Select

User ID: 1000

★ Boot Command: 1 sleep 60m

## Keeping a Failed Job Running

When creating a training job, add **|| sleep 5h** at the end of the boot command and start the training job. For example:

```
cmd || sleep 5h
```

If the training fails, the **sleep** command is executed. In this case, you can log in to the container image through Cloud Shell for debugging.

 **NOTE**

To debug a multi-node training job in Cloud Shell, you need to switch between worker-0 and worker-1 in Cloud Shell and run the boot command on each node. Otherwise, the task will wait for other nodes to join.

## Preventing Cloud Shell Session from Disconnection

To run a job for a long time, you can use the **screen** command to run the job in a remote terminal that stays active even if you disconnect. This prevents the job from failing due to disconnection.

1. If **screen** is not installed in the image, run **apt-get install screen** to install it.
2. Create a screen terminal.  
# Use **-S** to create a screen terminal named **name**.  

```
screen -S name
```
3. View the created screen terminals.  

```
screen -ls
```

There are screens on:

|                  |                       |            |
|------------------|-----------------------|------------|
| 2433.pts-3.linux | (2013-10-20 16:48:59) | (Detached) |
| 2428.pts-3.linux | (2013-10-20 16:48:05) | (Detached) |
| 2284.pts-3.linux | (2013-10-20 16:14:55) | (Detached) |
| 2276.pts-3.linux | (2013-10-20 16:13:18) | (Detached) |

4 Sockets in /var/run/screen/S-root.
4. Connect to the screen terminal whose **screen\_id** is **2276**.  

```
screen -r 2276
```
5. Press **Ctrl+A+D** to exit the screen terminal. After the exit, the screen session is still active and can be reconnected at any time.

For details about how to use screens, see [Screen User's Manual](#).

## Analyzing the Call Stack of the Suspended Process Using the py-spy Tool

Use py-spy to analyze the call stack of a suspended process and identify the issue.

**Step 1** Log in to the [ModelArts console](#). Choose **Model Training > Training Jobs**.

**Step 2** Click the target training job to go to its details page. On the page that appears, click the **Cloud Shell** tab and log in to the training container (the training job must be in the **Running** state).

**Step 3** Install the py-spy tool.

```
Use the utils.sh script to automatically configure the Python environment.
source /home/ma-user/modelarts/run/utils.sh

Install py-spy.
pip install py-spy

If the message "connection broken by 'ProxyError('Cannot connect to proxy:')" is displayed, disable the
proxy.
export no_proxy=$no_proxy,repo.myhuaweicloud.com (Replace it with the pip source address of the
corresponding region.)
pip install py-spy
```

**Step 4** View the stack. For details about how to use the py-spy tool, see the [py-spy official document](#).

```
Find the PID of the training process.
ps -ef

Check the process stack of process 12345.
```

```
For a training job using eight cards, run the following command to check the stacks of the eight processes started by the main process in sequence.
py-spy dump --pid 12345
```

----End

### 7.13.9 Saving the Image of a Debug Training Job

During development and debugging, you may modify and optimize the image environment. After modifying the image, you can save the image on the console for future use.

Notes for saving an image:

1. The installed dependency packages will be retained, but the training data and code that need to be persistently stored will not be saved in the generated container image.
2. VS Code keeps the plug-ins you install on the server during remote development.

Key settings and dependencies are preserved, but dynamic data and code require separate handling.

#### Prerequisites

- The debug job is created in a dedicated resource pool.
- The job is in the **Running** state.

#### Notes and Constraints

- Only the image of the training job's worker-0 can be saved on the console.
- The image to be saved should not be larger than 35 GB and there should be no more than 125 layers. Otherwise, the image may fail to be saved due to the rootfs difference between node containers.

#### Procedure

1. Click the target training job to access its details page. On the displayed page, click **Save Image** in the upper right corner.
2. In the **Save Image** dialog box, configure parameters. Click **OK** to save the image.

Choose an organization from the **Organization** drop-down list. If no organization is available, click **Create** on the right to create one.

Users in an organization can share all images in the organization.

The image will be saved as a snapshot, which will take a while. During which, the image status is **Saving**.

#### NOTE

The time required for saving an image as a snapshot will be counted in the job runtime. If the job runtime expires before the snapshot is saved, saving the image will fail.

3. The saved image can be used to create a training job.

## FAQs

1. What Can I Do If an Image Fails to Be Saved?
  - a. If the image fails to be saved, check the event on the training job details page. For details, see [Viewing Training Job Events](#).
  - b. If a dedicated resource pool is used, you can adjust the container engine space on the **Resource Management > Lite Cluster** page as required. For details, see [Changing the Number of Instances](#).
  - c. If the problem still persists, contact technical engineers.
2. Why Cannot I Check Real-Time Logs During Image Saving?

Saving an image may affect checking real-time logs. The Cloud Shell connection may be interrupted. After the image is saved, the connection is automatically restored.

## 7.13.10 Copying, Stopping, or Deleting a Training Job

### Saving As an Algorithm

To modify the algorithm of a training job, click **Save As Algorithm** in the upper right corner of the training job details page.

On the **Algorithms** page, the algorithm parameters for the last training job are automatically set. You can modify the settings.

#### NOTE

A subscribed algorithm cannot be saved as a new algorithm.

### Copying a Training Job

If you are not satisfied with a created training job, click **Copy** in the **Operation** column to copy it. The page for creating a training job is displayed. On this page, the parameter settings for the previous training job are automatically retained. You only need to modify certain parameter settings.

### Stopping a Training Job

In the training job list, click **Stop** in the **Operation** column of a training job that is in creating, pending, or running state. Enter **YES** in the dialog box and confirm the operation.

After a training job is stopped, its billing stops on ModelArts.

A training job in completed, failed, terminated, or abnormal state cannot be stopped.

### Deleting a Training Job

Release resources of a training job when not in use to avoid unnecessary charges.

#### NOTE

Deleted training jobs cannot be restored.

- On the **Training Jobs** page, Click **Delete** in the **Operation** column. In the displayed dialog box, enter **DELETE**, and click **OK** to confirm the deletion.
- Go to OBS and delete the OBS bucket and files used by the training job.

## Searching for a Training Job

If you log in to ModelArts using an IAM account, all training jobs under this account are displayed in the training job list. To quickly search for a training job, use the following methods:

Method 1: Click **Only my jobs**. Then, only jobs created under the current IAM user account are displayed in the training job list.

Method 2: Search for jobs by name, ID, job type, status, creation time, algorithm, and resource pool.

Method 3: Click the refresh button in the upper right corner of the job list to refresh it.

Method 4: Configure the custom columns and other basic settings.

**Figure 7-52** Searching for a training job



## 7.13.11 Managing Environment Variables of a Training Container

### What Is an Environment Variable

This section describes environment variables preset in a training container. The environment variables include:

- Path environment variables
- Environment variables of a distributed training job
- Nvidia Collective multi-GPU Communication Library (NCCL) environment variables
- OBS environment variables
- Environment variables of the pip source
- Environment variables of the API Gateway address
- Environment variables of job metadata

### Notes and Constraints

When defining custom environment variables, avoid using names that start with **MA\_** to prevent conflicts with system environment variables.

### Configuring Environment Variables

When you create a training job, you can add environment variables or modify environment variables preset in the training container.



 NOTE

To ensure data security, do not enter sensitive information, such as plaintext passwords.

## Environment Variables Preset in a Training Container

[Table 7-48](#), [Table 7-49](#), [Table 7-50](#), [Table 7-51](#), [Table 7-52](#), [Table 7-53](#), and [Table 7-54](#) list environment variables preset in a training container.

The environment variable values are examples only.

**Table 7-48** Path environment variables

| Variable              | Description                                                                                                                                         | Example                                                                                                                                                           |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PATH                  | Executable file paths                                                                                                                               | <code>PATH=/usr/local/bin:/usr/local/cuda/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin</code>                                                 |
| LD_LIBRARY_PATH       | Dynamic load library paths                                                                                                                          | <code>LD_LIBRARY_PATH=/usr/local/seccomponent/lib:/usr/local/cuda/lib64:/usr/local/cuda/compat:/root/miniconda3/lib:/usr/local/lib:/usr/local/nvidia/lib64</code> |
| LIBRARY_PATH          | Static library paths                                                                                                                                | <code>LIBRARY_PATH=/usr/local/cuda/lib64/stubs</code>                                                                                                             |
| MA_HOME               | Main directory of a training job                                                                                                                    | <code>MA_HOME=/home/ma-user</code>                                                                                                                                |
| MA_JOB_DIR            | Parent directory of the training algorithm folder                                                                                                   | <code>MA_JOB_DIR=/home/ma-user/modelarts/user-job-dir</code>                                                                                                      |
| MA_MOUNT_PATH         | Path mounted to a ModelArts training container, which is used to temporarily store training algorithms, algorithm input, algorithm output, and logs | <code>MA_MOUNT_PATH=/home/ma-user/modelarts</code>                                                                                                                |
| MA_LOG_DIR            | Training log directory                                                                                                                              | <code>MA_LOG_DIR=/home/ma-user/modelarts/log</code>                                                                                                               |
| MA_SCRIPT_INTERPRETER | Training script interpreter                                                                                                                         | <code>MA_SCRIPT_INTERPRETER=</code>                                                                                                                               |
| WORKSPACE             | Training algorithm directory                                                                                                                        | <code>WORKSPACE=/home/ma-user/modelarts/user-job-dir/code</code>                                                                                                  |

**Table 7-49** Environment variables of a distributed training job

| Variable      | Description                                                                                                                                                                                                                                                   | Example                            |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------|
| MA_CURRENT_IP | IP address of a job container.                                                                                                                                                                                                                                | <b>MA_CURRENT_IP=192.168.23.38</b> |
| MA_NUM_GPUS   | Number of accelerator cards in a job container.                                                                                                                                                                                                               | <b>MA_NUM_GPUS=8</b>               |
| MA_TASK_NAME  | Name of a job container, for example: <ul style="list-style-type: none"><li>• <b>worker</b> in MindSpore and PyTorch</li><li>• <b>learner</b> or <b>worker</b> in reinforcement learning engines</li><li>• <b>ps</b> or <b>worker</b> in TensorFlow</li></ul> | <b>MA_TASK_NAME=worker</b>         |
| MA_NUM_HOSTS  | Number of instances which is automatically obtained from <b>Compute Nodes</b> .                                                                                                                                                                               | <b>MA_NUM_HOSTS=4</b>              |
| VC_TASK_INDEX | Container index, starting from <b>0</b> . This parameter is invalid for single-node training. In multi-node training jobs, you can use this parameter to determine the algorithm logic of the container.                                                      | <b>VC_TASK_INDEX=0</b>             |
| VC_WORKER_NUM | Instances required for a training job.                                                                                                                                                                                                                        | <b>VC_WORKER_NUM=4</b>             |

| Variable                                                                                                                          | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | Example                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-----------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| VC_WORKER_HOSTS                                                                                                                   | Domain name of each node for multi-node training. Use commas (,) to separate the domain names in sequence. You can obtain the IP address through domain name resolution.                                                                                                                                                                                                                                                                                                                    | <b>VC_WORKER_HOSTS=modelarts-job-a0978141-1712-4f9b-8a83-000000000000-worker-0.modelarts-job-a0978141-1712-4f9b-8a83-000000000000,modelarts-job-a0978141-1712-4f9b-8a83-000000000000-worker-1.ob-a0978141-1712-4f9b-8a83-000000000000,modelarts-job-a0978141-1712-4f9b-8a83-000000000000-worker-2.modelarts-job-a0978141-1712-4f9b-8a83-000000000000,ob-a0978141-1712-4f9b-8a83-000000000000-worker-3.modelarts-job-a0978141-1712-4f9b-8a83-000000000000</b> |
| <code>\$</code><br><code>{MA_VJ_NAME}</code><br><code>-\$</code><br><code>{MA_TASK_NAME}-N.\$</code><br><code>{MA_VJ_NAME}</code> | Communication domain name of a node. For example, the communication domain name of node 0 is <code>\$ {MA_VJ_NAME}-\$ {MA_TASK_NAME}-0.\$ {MA_VJ_NAME}</code> .<br>N indicates the number of instances.<br><b>WARNING</b><br>This method does not work for creating communication domain names for supernode resource pools.<br>Instead, you can get the communication domain names for all nodes directly from <b>VC_WORKER_HOSTS</b> across all resource pools, including supernode ones. | For example, if there are four instances, the environment variables are as follows:<br><code>{MA_VJ_NAME}-\$ {MA_TASK_NAME}-0.\$ {MA_VJ_NAME}</code><br><code>{MA_VJ_NAME}-\$ {MA_TASK_NAME}-1.\$ {MA_VJ_NAME}</code><br><code>{MA_VJ_NAME}-\$ {MA_TASK_NAME}-2.\$ {MA_VJ_NAME}</code><br><code>{MA_VJ_NAME}-\$ {MA_TASK_NAME}-3.\$ {MA_VJ_NAME}</code>                                                                                                      |

**Table 7-50** NCCL environment variables

| Variable     | Description  | Example                   |
|--------------|--------------|---------------------------|
| NCCL_VERSION | NCCL version | <b>NCCL_VERSION=2.7.8</b> |

| Variable           | Description                                       | Example                               |
|--------------------|---------------------------------------------------|---------------------------------------|
| NCCL_DEBUG         | NCCL log level                                    | <b>NCCL_DEBUG=INFO</b>                |
| NCCL_IB_HCA        | InfiniBand NIC to use for communication           | <b>NCCL_IB_HCA=^mlx5_bond_0</b>       |
| NCCL_IB_TIMEOUT    | InfiniBand transmission timeout interval          | <b>NCCL_IB_TIMEOUT=18</b>             |
| NCCL_IB_RETRY_CNT  | Maximum number of InfiniBand transmission retries | <b>NCCL_IB_RETRY_CNT=15</b>           |
| NCCL_IB_GID_INDEX  | Global ID index used in RoCE mode                 | <b>NCCL_IB_GID_INDEX=3</b>            |
| NCCL_IB_TC         | InfiniBand traffic type                           | <b>NCCL_IB_TC=128</b>                 |
| NCCL_SOCKET_IFNAME | IP interface to use for communication             | <b>NCCL_SOCKET_IFNAME=bond0, eth0</b> |
| NCCL_NET_PLUGIN    | Network plug-in used by NCCL                      | <b>NCCL_NET_PLUGIN=none</b>           |

**Table 7-51** OBS environment variables

| Variable       | Description                        | Example                |
|----------------|------------------------------------|------------------------|
| MA_S3_ENDPOINT | OBS endpoint                       | N/A                    |
| S3_VERIFY_SSL  | Whether to use SSL to access OBS   | <b>S3_VERIFY_SSL=0</b> |
| S3_USE_HTTPS   | Whether to use HTTPS to access OBS | <b>S3_USE_HTTPS=1</b>  |

**Table 7-52** Environment variables of the pip source and API Gateway address

| Variable          | Description                   | Example                                                           |
|-------------------|-------------------------------|-------------------------------------------------------------------|
| MA_PIP_HOST       | Domain name of the pip source | <b>MA_PIP_HOST=repo.example.com</b>                               |
| MA_PIP_URL        | Address of the pip source     | <b>MA_PIP_URL=http://repo.example.com/repository/pypi/simple/</b> |
| MA_APIGW_ENDPOINT | ModelArts API Gateway address | <b>MA_APIGW_ENDPOINT=https://modelarts.region.xxx.example.com</b> |

**Table 7-53** Environment variables of job metadata

| Variable                 | Description                                      | Example                                                                                   |
|--------------------------|--------------------------------------------------|-------------------------------------------------------------------------------------------|
| MA_CURRENT_INSTANCE_NAME | Name of the current node for multi-node training | <b>MA_CURRENT_INSTANCE_NAME=modelarts-job-a0978141-1712-4f9b-8a83-0000000000-worker-1</b> |

**Table 7-54** Precheck environment variables

| Variable             | Description                                                                                                                                                                                                                                                                                   | Example  |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------|
| MA_SKIP_IMAGE_DETECT | Whether to enable ModelArts precheck. The default value is <b>1</b> , which indicates that the pre-check is enabled; the value <b>0</b> indicates that the pre-check is disabled.<br><br>It is good practice to enable precheck to detect node and driver faults before they affect services. | <b>1</b> |

**Table 7-55** Suspension detection environment variables

| Variable            | Description                                                                                                                                                                            | Example   |
|---------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------|
| MA_HANG_DETECT_TIME | Suspension detection time. The job is considered suspended if its process I/O does not change for this time.<br><br>Value range: 10 to 720<br>Unit: minute<br>Default value: <b>30</b> | <b>30</b> |

## How Do I View Training Environment Variables?

Environment variables may be injected into the container or processes, depending on the service. Environment variables injected into processes are invisible in Cloud Shell.

You are advised to use method [1](#) to view all environment variables.

1. View training environment variables using the boot command.

When creating a training job, select **Custom algorithm** for **Algorithm Type** and **Custom image** for **Boot Mode**, enter **env** for **Boot Command**, and retain default settings for other parameters.

Figure 7-53 Boot Command

Basic Configuration

Environment Settings

Training Configuration

Resource configuration

Fault tolerance and recovery

Scheduling Settings

More Configurations

Environment Settings

Algorithm Type ?

Custom algorithm My algorithm My subscription

Boot Mode ?

Preset image Custom image

Image ?

Select an image.

☒ Code Source

OBS

Code Directory (Optional) ?

Select a code directory.

Boot Command

1 env

After the training job is complete, check the **Logs** tab on the training job details page. The logs contain information about all environment variables.

Figure 7-54 Viewing logs

```
1 NV_LIBCUBLAS_DEV_VERSION=11.3.1.68-1
2 NV_CUDA_COMPAT_PACKAGE=cuda-compat-11-2
3 NV_CUDNN_PACKAGE_DEV=libcudnn8-dev=8.1.1.33-1+cuda11.2
4 LD_LIBRARY_PATH=/usr/local/nvidia/lib:/usr/local/nvidia/lib64
5 NV_LIBNCCL_DEV_PACKAGE=libnccl-dev=2.8.4-1+cuda11.2
6 MA_ENGINE_VERSION=
7 MA_NUM_HOSTS=1
8 VC_WORKER_HOSTS=modelarts-job-5f8e4b52-630b-4c15-9bb6-c3f68a48ac47-worker-0.modelarts-job-5f8e4b52-630b-4c15-9bb6-c3f68a48ac47
9 VK_TASK_INDEX=0
10 _=/usr/bin/env
11 MA_SCRIPT_INTERPRETER=
12 NV_LIBNPP_DEV_PACKAGE=libnpp-dev-11-2=11.2.1.68-1
13 MA_MAX_BACKOFF=0
14 HOSTNAME=modelarts-job-5f8e4b52-630b-4c15-9bb6-c3f68a48ac47-worker-0
15 MA_IAM_USER_ID=79098163dd814fd986eca7ef0325d086
16 MA_CURRENT_IP=10.0.0.62
17 NV_LIBNPP_VERSION=11.2.1.68-1
18 NV_NVPROF_DEV_PACKAGE=cuda-nvprof-11-2=11.2.67-1
19 MA_MOUNT_PATH=/home/ma-user/modelarts
20 NVIDIA_VISIBLE_DEVICES=all
21 MA_ENGINE_TYPE=
22 NV_NVPROF_VERSION=11.2.67-1
23 NV_LIBCUSPARSE_VERSION=11.3.1.68-1
24 MODELARTS_SCC_SERVICE_PORT=60687
25 MA_HOME=/home/ma-user
26 KUBERNETES_PORT_443_TCP_PROTO=tcp
27 KUBERNETES_PORT_443_TCP_ADDR=10.247.0.1
28 NV_LIBCUBLAS_DEV_PACKAGE=libcublas-dev-11-2=11.3.1.68-1
29 MA_VJ_NAME=modelarts-job-5f8e4b52-630b-4c15-9bb6-c3f68a48ac47
30 MA_PIP_URL=http://repo.myhuaweicloud.com/repository/pypi/simple/
31 NCCL_VERSION=2.8.4-1
32 KUBERNETES_PORT=tcp://10.247.0.1:443
33 PWD=/
34 NARCH=x86_64
35 HOME=/home/ma-user
```

2. Check training environment variables using Cloud Shell.  
Run the **env** command on Cloud Shell to obtain the environment variables.  
This method fails to obtain environment variables injected by the training platform during processes, such as **VC\_TASK\_INDEX**, **VC\_WORKER\_NUM**, and **VC\_WORKER\_HOSTS** in supernode scenarios. This method is not recommended.

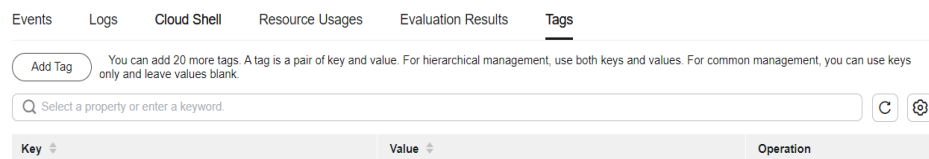
## 7.13.12 Viewing Training Job Tags

You can add tags to a training job for quick search.

1. On the [ModelArts console](#), choose **Model Training** > **Training Jobs** from the navigation pane.
2. In the training job list, click the name of the target job to go to the training job details page.
3. Click **Tags**.

Tags can be added, modified, and deleted. For details about how to use tags, see [Using TMS Tags to Manage Resources by Group](#).

**Figure 7-55** Viewing training tags



### NOTE

You can add up to 20 tags to a training job.

## 7.13.13 Managing Training Experiments

### Training Experiment

Managing many training jobs can become challenging when trying to find or track them. Training experiments can simplify this process. This is a way to group similar jobs together. You can organize jobs into different experiments as needed, with each experiment holding multiple related jobs.

You can add one job to an experiment, create jobs within existing experiments, check the experiment list, view experiment details, or delete experiments as needed.

### Adding a Training Job to an Experiment

To add a training job to an experiment, configure **Experiment** when creating a training job. The options are as follows:

- **Create new:** An experiment can only be created when you create a training job. If you select this option, enter a new experiment name. After the job is submitted, the experiment is created and the job is added to the new

experiment. The experiment name will be checked. If the name is already in use, the job cannot be submitted.

- **Use existing:** Select an existing experiment from the drop-down list box to add the job to the existing experiment.
- **Not required:** Select this option if you do not want to manage your job through an experiment. The experiment tab on the training job management page only shows jobs that have been added to an experiment.

## Creating a Job to Be Added to an Experiment

Log in to the [ModelArts console](#), choose **Model Training > Training Jobs**, and click **Create Training Job** in the upper right corner.

On this page, set **Experiment** to **Create new** and enter a name for the new experiment. Then, an experiment is created after you create the training job.

**Figure 7-56** Creating a training job

☒ Lab Setup

Enable this feature to organize training jobs into experiments for better management. It helps manage multiple job versions efficiently.

Use existing Create new

Experiment Name

Please enter an experiment name.

Experiment Description (Optional)

Please enter the lab description.

0/256

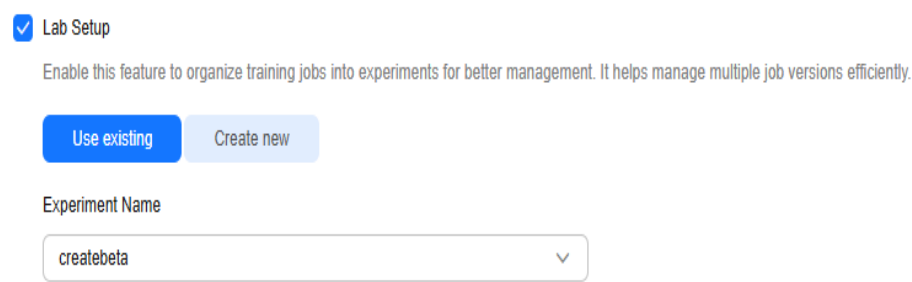
## Adding a Created Job to an Experiment

Log in to the [ModelArts console](#), choose **Model Training > Training Jobs**, and click **Copy** in the **Operation** column of the target job. Alternatively, click the job name or ID in the job list. On the job details page, click **Copy** in the upper right corner.

- For a job that has not been added to an experiment, set **Experiment** to **Create new** and enter a name for the new experiment. Then, an experiment is created after you copy the training job.
- For a job that has been added to an experiment, set **Experiment** to **Use existing** and select the target experiment.



Figure 7-57 Copying a training job



Viewing the Experiment List

- 1. Log in to the **ModelArts console**. In the navigation pane on the left, choose **Model Training > Training Jobs** to go to the **Training Jobs** page.
- 2. Click **Experiments**. The experiment list displays some basic experiment information.

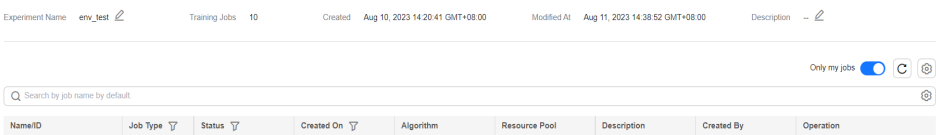
Table 7-56 Basic experiment information

| Parameter       | Description                                                                                                                                                                                                                                                   |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Experiment Name | Experiment name, which can be changed on the experiment details page                                                                                                                                                                                          |
| Training Jobs   | Number of training jobs in an experiment                                                                                                                                                                                                                      |
| Created         | Time when an experiment is created                                                                                                                                                                                                                            |
| Modified At     | Time when any of the following occurs: <ul style="list-style-type: none"><li>• Changing the experiment name</li><li>• Modifying the description of the experiment</li><li>• Adding a training job to or deleting a training job from the experiment</li></ul> |
| Description     | Experiment description, which can be modified                                                                                                                                                                                                                 |
| Operation       | You can delete the experiment.                                                                                                                                                                                                                                |

Viewing Experiment Details

In the experiment list, click an experiment name to go to the experiment details page. Basic experiment information is displayed in the upper part of the experiment details page, and the job list of the experiment is displayed in the lower part of the experiment details page.

Figure 7-58 Viewing experiment details



- You can click  to edit the name and description of an experiment.
- You can click **Only my jobs** to view the jobs that you have created and included in the experiment.

 **NOTE**

By default, if an account has multiple IAM users, only the jobs of the current IAM user is displayed.

- You can search for jobs by name, ID, algorithm, status, creation time, job type, or resource pool.
- You can refresh the job list. You can click the refresh button in the upper right corner of the job list to refresh it.
- You can click the setting button in the upper right corner of the job list to select items you want to display in the job list.

## Deleting an Experiment

 **WARNING**

After an experiment is deleted, all jobs in the experiment will be deleted accordingly and cannot be restored. Therefore, exercise cautions when performing this operation.

You can click **Delete** on the experiment list page or click **Delete Experiment** in the upper right corner of the experiment details page to delete an experiment. All jobs of the experiment are displayed on the **Delete Experiment** page. Enter **DELETE** and click **OK** to confirm the deletion.

### 7.13.14 Viewing Monitoring Metrics of a Training Job

Receiving and promptly addressing alarms during a training job (for example, abnormal loss values) can save significant time and resources, preventing the waste caused by invalid job runs. Additionally, metric monitoring allows you to track the training job's progress in real time and the model's training status across different phases.

In the **Monitoring** tab of the training job details page of ModelArts Standard, you can view the CPU, GPU, or NPU usage of training jobs. For details, see [Viewable Metrics on the ModelArts Console](#).

For more metrics, go to the [AOM console](#). For details, see [Viewing All ModelArts Monitoring Metrics on the AOM Console](#).

With ModelArts, you can track custom metrics in AOM, such as loss values, step durations, and GPU throughput. These metrics are displayed in your training logs, making it easy to monitor trends and compare results across different jobs.

- **Reporting Custom Metrics to AOM Using SDK:** Integrate the SDK into your code to upload metric data to AOM manually.
  - This method works well for users who need to add custom metrics, like adding custom dimensions or complex calculations.
  - This method works well for complex service logic and when you need to control metric reporting in your code.

- This method works well for users who have experience with Huawei Cloud SDKs and know about cloud services.

## Viewable Metrics on the ModelArts Console

**Table 7-45** shows the metrics that can be viewed on the training job details page of the **ModelArts console**.

**Table 7-57** shows the training metrics that can be viewed on the **Overview** page of the **ModelArts console**.

**Table 7-57** Task-level metrics of training jobs

| Metric                      | Description                                                  | How to View                                                                                                                                                                |
|-----------------------------|--------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Training job resource usage | CPU, GPU, or NPU usage of a training job.                    | <b>ModelArts console</b> > <b>Overview</b> > <b>Resource Usage of Training Jobs</b>                                                                                        |
| Card-hours                  | Running duration and number of cards used by a training job. | <b>ModelArts console</b> > <b>Overview</b> > <b>Resource Usage of Training Jobs</b><br> |

## Reporting Custom Metrics to AOM Using SDK

Integrate the SDK into your code to upload metric data to AOM manually.

This method works well for users who need to add custom metrics, like adding custom dimensions or complex calculations. It also works well for complex service logic and when you need to control metric reporting in your code.

1. Add the metric monitoring code to the training code. You can refer to the sample code below. For details about other requirements for preparing training code, see **Preparing Model Training Code**.

Replace the region value **cn-southwest-2** in the last but one line of the code with the actual region. For details about the region values, see **Endpoints**.

Add monitoring metrics to the code. For details about the parameters, see the **AOM documentation**.

```
coding: utf-8
import os

from huaweicloudsdaom.v2 import *
from huaweicloudsdaom.v2.region.aom_region import AomRegion
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.exceptions import exceptions
from moxing.framework import cloud_utils

def report2Aom(request,region):
 auth = cloud_utils.get_auth() # AK, SK, and temporary token, which are automatically obtained by
```

the system.

```
ak = auth.AK
sk = auth.SK
securityToken = auth.TOKEN

projectId = os.environ.get("MA_IAM_PROJECT_ID")
credentials = BasicCredentials(ak, sk, projectId).with_security_token(securityToken)
client = AomClient.new_builder() \
 .with_credentials(credentials) \
 .with_region(AomRegion.value_of(region)) \
 .build()
try:
 response = client.add_metric_data(request)
 print(response)
except Exception as e:
 print(e)
if __name__ == "__main__":

 request = AddMetricDataRequest()
 listValuesBody = [
 # Enter the metric name, type, unit, and value, such as step_time and loss value.
 ValueData(
 metric_name="step_time", # Monitoring metric name, for example, step_time
 type="float", # Data type of the metric. The value can only be int or float.
 unit="ms", # Data unit, for example, ms. The value contains a maximum of 32 characters.
 value=135.572 # Value of the metric data. The value must be of a valid numeric type. The
minimum value is 0.
),
 ValueData(
 metric_name="loss",
 type="float",
 value=0.6932
)
]
 listDimensionsMetric = [
 # Enter the metric dimensions you want to view, such as thread and host.
 Dimension2(
 name="cluster_name",# This is only an example. Replace it with the actual metric dimension
you want to view.
 value="fab2c5cf438b4f0c851fdcdf"# This is only an example. Replace it with the actual
parameter value.
),
 Dimension2(
 name="user_name",
 value="modelarts_02" # This is only an example. Replace it with the actual parameter value.
),
 Dimension2(
 name="user_id",
 value="04f258c8fb00d42a1f6xxx" # This is only an example. Replace it with the actual
parameter value.
)
]
 metricBody = MetricItemInfo(
 dimensions=listDimensionsMetric,
 namespace="NOPAAS.ESC" # Retain the default value.
)
 listBodybody = [
 MetricDataItem(
 collect_time=int(round(time.time()*1000)), # Time when monitoring metric data is collected.
The value is the latest timestamp, in milliseconds.
 metric=metricBody,
 values=listValuesBody
)
]
 request.body = listBodybody
 region = "cn-southwest-2" # Replace the value with the actual region.
 response = report2Aom(request,region)
```

2. Add the commands below to the training code to load the required dependency packages. If a custom image is used, you can also install the dependencies during image creation. For details, see [Developing Code for Training Using a Custom Image](#).  

```
pip install huaweicloudsdfaom
pip install huaweicloudsdkscore
```
3. Create and run a training job. For details, see [Creating a Production Training Job](#).
4. Log in to the [AOM console](#). On the **Metric Browsing** page, view the reported metric data by specifying metrics.
5. Set the AOM alarm and notification mechanisms by referring to [Configuring Alarm Settings](#).

# 8 Using ModelArts Standard to Deploy Models for Inference and Prediction

---

## 8.1 Overview

Inference deployment is an important part of AI and machine learning. Inference deployment moves a trained machine learning or deep learning model from the development environment to the production environment. This allows the model to predict outcomes for new, unseen data. The goal is to ensure the model works efficiently and reliably in real-world applications, meeting real-time and resource needs. Simply put, it makes the model operational, providing outputs like image classification, text sentiment analysis, and trend predictions based on input data.

ModelArts offers you compute, storage, and network resources to deploy, use, manage, and monitor your AI model after development. ModelArts supports cloud, edge, and on-device deployments. It also supports real-time, batch, and edge inference.

### Deployment Modes

ModelArts supports cloud deployments. It also supports real-time and batch inference.

Cloud-based deployment means deploying and running inference services on cloud servers. This works well for tasks needing lots of compute and data.

- **Real-time inference:** Processes a single request instantly and returns the result right away. ModelArts allows you to deploy a model as a web service that provides a real-time test UI and monitoring capabilities. This service provides a callable API. Real-time inference is used in situations that need fast responses, like online intelligent customer service and autonomous driving decisions.
- **Batch inference:** Processes multiple inputs at once and returns all results together. ModelArts allows you to deploy a model as a batch service. This service runs inference on batch data and stops automatically when done. Batch inference works well for offline tasks like big data analysis, batch data labeling, and model evaluation.

## Description

Deploying a service in ModelArts uses compute and storage resources, which are billed. Compute resources are billed for running the inference service. Storage resources are billed for storing data in OBS. For details, see [Table 8-1](#).

**Table 8-1** Billing items

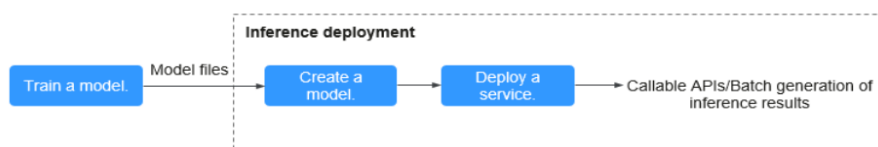
| Billing Item                                  |                              | Description                                                                                                                                                                                                                                                                                  | Billing Mode                      | Billing Formula                                                                                                                                                                                                          |
|-----------------------------------------------|------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Compute resource                              | Public resource pools        | Usage of compute resources.<br>For details, see <a href="#">ModelArts Pricing Details</a> .                                                                                                                                                                                                  | Pay-per-use                       | Specification unit price x Number of compute nodes x Usage duration                                                                                                                                                      |
|                                               | Dedicated resource pools     | Fees for dedicated resource pools are paid upfront upon purchase. There are no additional charges for service deployment.<br>For details about dedicated resource pool fees, see <a href="#">Dedicated Resource Pool Billing Items</a> .                                                     | N/A                               | N/A                                                                                                                                                                                                                      |
| Storage resource                              | Object Storage Service (OBS) | OBS is used to store the input and output data for service deployment.<br>For details, see <a href="#">Object Storage Service Pricing Details</a> .<br><b>CAUTION</b><br>OBS resources for storing data are continuously billed. To stop billing, delete the data stored in OBS.             | Pay-per-use<br>Yearly/<br>Monthly | Creating an OBS bucket is free of charge. You pay only for the storage capacity and duration you actually use.                                                                                                           |
| Event notification (billed only when enabled) |                              | This function uses Simple Message Notification (SMN) to send a message to you when the event you selected occurs.<br>To use this function, enable event notification when creating a training job.<br>For pricing details, see <a href="#">Simple Message Notification Pricing Details</a> . | Pay by actual usage               | <ul style="list-style-type: none"><li>SMS: SMS notifications</li><li>Email: Email notifications + Downstream Internet traffic</li><li>HTTP or HTTPS: HTTP or HTTPS notifications + Downstream Internet traffic</li></ul> |

| Billing Item                        | Description                                                                                                                                                                                                                                                                     | Billing Mode           | Billing Formula                                                                                         |
|-------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|---------------------------------------------------------------------------------------------------------|
| Run logs (billed only when enabled) | <b>Log Tank Service (LTS)</b> collects, analyzes, and stores logs.<br><br>If <b>Runtime Log Output</b> is enabled during service deployment, you will be billed if the log data exceeds the LTS free quota. For details, see <a href="#">Log Tank Service Pricing Details</a> . | Pay by actual log size | After the free quota is exceeded, you are billed based on the actual log volume and retention duration. |

## Inference Deployment Process

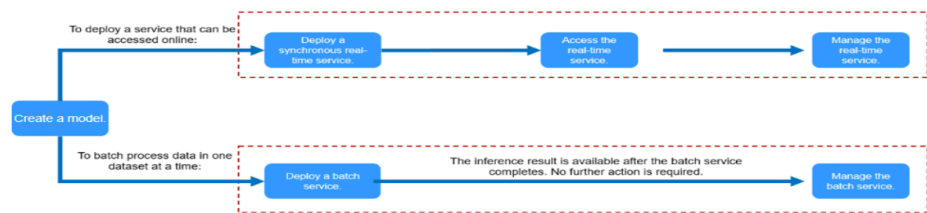
You can import and deploy AI models as inference services. These services can be integrated into your IT platform by calling APIs or generate batch results.

**Figure 8-1** Introduction to inference



1. Prepare inference resources: Select required resources based on the site requirements. ModelArts provides you with public resource pools and dedicated resource pools. To use dedicated compute resources, you need to purchase and create a dedicated resource pool first. For details, see [Creating a Dedicated Resource Pool](#).
2. Train a model: Models can be trained in ModelArts or your local development environment. A locally developed model must be uploaded to Huawei Cloud OBS.
3. Create a model: Import the model file and inference file to the ModelArts model repository and manage them by version. Use these files to build an executable model.
4. Deploy a service: Deploy the model as a service type based on your needs.
  - [Deploying a Model as Real-Time Inference Jobs](#)  
Deploy a model as a web service with real-time UI and monitoring. This service provides you a callable API.
  - [Deploying a Model as a Batch Inference Service](#)  
Deploy an AI application as a batch service that performs inference on batch data and automatically stops after data processing is complete.



**Figure 8-2** Different inference scenarios

## 8.2 Creating a Model

### 8.2.1 Creation Methods

AI development and optimization require frequent iterations and debugging. Modifications in datasets, training code, or parameters affect the quality of models. If the metadata of the development process cannot be centrally managed, the optimal model may fail to be reproduced.

With ModelArts, you can create models using meta models from training jobs, OBS, or container images, and centrally manage all iterated and debugged models.

#### Constraints

- After deploying a model in an ExeML project, it is automatically added to the model list. ExeML-generated models can only be deployed, not downloaded.
- All users can create models and manage model versions at no cost.

#### Meta Model Sources

- **Importing a Meta Model from a Training Job:** Create a training job in ModelArts to train a model. After obtaining a desired model, use it to create a model for service deployment.
- **Importing a Meta Model from OBS:** If you use a mainstream framework to develop and train a model locally, you can upload the model to an OBS bucket based on the model package specifications, import the model from OBS to ModelArts, and use the model for service deployment.
- **Importing a Meta Model from a Container Image:** If an AI engine is not supported by ModelArts, you can use it to build a model, import the model to ModelArts as a custom image, and use the image to create a model for service deployment.

#### Supported AI Engines for ModelArts Inference

If you import a model from OBS to ModelArts, the following AI engines and versions are supported.

 NOTE

- A runtime environment of a unified image is named in the following format: *<AI engine and version> - <Hardware and version: CPU, CUDA, or CANN> - <Python version> - <OS version> - <CPU architecture>*
- Each preset AI engine has its default model start command. Do not modify it unless necessary.

**Table 8-2** Supported AI engines, their runtime environments, and default start commands

| Engine       | Runtime Environment                                                                                                                                                                                                                                    | Note                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TensorFlow   | python3.6<br>python2.7 (unavailable soon)<br>tf1.13-python3.6-gpu<br>tf1.13-python3.6-cpu<br>tf1.13-python3.7-cpu<br>tf1.13-python3.7-gpu<br>tf2.1-python3.7 (unavailable soon)<br>tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64 (recommended) | <ul style="list-style-type: none"><li>• TensorFlow 1.8.0 is used in <b>python2.7</b> and <b>python3.6</b>.</li><li>• The model can run on both CPUs and GPUs when using <b>python3.6</b>, <b>python2.7</b>, or <b>tf2.1-python3.7</b>. If the runtime environment has a suffix of <b>cpu</b> or <b>gpu</b>, the model can only run on CPUs or GPUs respectively.</li><li>• The default runtime environment is <b>python2.7</b>.</li><li>• Default start command: <b>sh / home/mind/run.sh</b></li></ul> |
| Spark_MLlib  | python2.7 (unavailable soon)<br>python3.6 (unavailable soon)                                                                                                                                                                                           | <ul style="list-style-type: none"><li>• Spark_MLlib 2.3.2 is used in <b>python2.7</b> and <b>python3.6</b>.</li><li>• The default runtime environment is <b>python2.7</b>.</li><li>• <b>python2.7</b> and <b>python3.6</b> can only be used to run models on CPUs.</li><li>• Default start command: <b>bash / home/work/predict/bin/run.sh</b></li></ul>                                                                                                                                                |
| Scikit_Learn | python2.7 (unavailable soon)<br>python3.6 (unavailable soon)                                                                                                                                                                                           | <ul style="list-style-type: none"><li>• Scikit_Learn 0.18.1 is used in <b>python2.7</b> and <b>python3.6</b>.</li><li>• The default runtime environment is <b>python2.7</b>.</li><li>• <b>python2.7</b> and <b>python3.6</b> can only be used to run models on CPUs.</li><li>• Default start command: <b>bash / home/work/predict/bin/run.sh</b></li></ul>                                                                                                                                              |

| Engine    | Runtime Environment                                                                                                                                                                                            | Note                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| XGBoost   | python2.7<br>(unavailable soon)<br>python3.6<br>(unavailable soon)                                                                                                                                             | <ul style="list-style-type: none"><li>• XGBoost 0.80 is used in <b>python2.7</b> and <b>python3.6</b>.</li><li>• The default runtime environment is <b>python2.7</b>.</li><li>• <b>python2.7</b> and <b>python3.6</b> can only be used to run models on CPUs.</li><li>• Default start command: <b>bash / home/work/predict/bin/run.sh</b></li></ul>                                                                                             |
| PyTorch   | python2.7<br>(unavailable soon)<br>python3.6<br>python3.7<br>pytorch1.4-python3.7<br>pytorch1.5-python3.7<br>(unavailable soon)<br>pytorch_1.8.0-<br>cuda_10.2-py_3.7-<br>ubuntu_18.04-x86_64<br>(recommended) | <ul style="list-style-type: none"><li>• PyTorch 1.0 is used in <b>python2.7</b>, <b>python3.6</b>, and <b>python3.7</b>.</li><li>• The model can run on both CPUs and GPUs when using <b>python2.7</b>, <b>python3.6</b>, <b>python3.7</b>, <b>pytorch1.4-python3.7</b>, or <b>pytorch1.5-python3.7</b>.</li><li>• The default runtime environment is <b>python2.7</b>.</li><li>• Default start command: <b>sh / home/mind/run.sh</b></li></ul> |
| MindSpore | aarch64<br>(recommended)                                                                                                                                                                                       | <p><b>aarch64</b> can only be used to run models on Snt3 chips.</p> <ul style="list-style-type: none"><li>• Default start command: <b>sh / home/mind/run.sh</b></li></ul>                                                                                                                                                                                                                                                                       |

## 8.2.2 Importing a Meta Model from a Training Job

Create a training job in ModelArts to obtain a satisfactory model. The model can then be imported to create an AI application for centralized management. The application can be quickly deployed as a service.

### Constraints

- You can directly import a model generated from a training job that uses a subscribed algorithm to ModelArts, without needing to use the inference code or configuration file.
- If the meta model is from a container image, ensure the size of the meta model complies with [Restrictions on the Size of an Image for Importing a Model](#).

## Prerequisites

- The training job has been executed, and the model has been stored in the OBS directory where the training output is stored (the input parameter is **train\_url**).
- If the training job uses a mainstream framework or custom image, upload the inference code and configuration file to the model storage directory by referring to [Model Package Structure](#).
- The OBS directory you use must be in the same region as ModelArts.

## Procedure

1. Log in to the ModelArts console and choose **Model Management** in the navigation pane on the left.
2. Click **Create Model**.
3. Configure parameters.
  - a. Set basic information about the model. For details about the parameters, see [Table 8-3](#).

**Table 8-3** Basic information

| Parameter   | Description                                                                                                                                                                                                  |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Name        | Model name. The value can contain 1 to 64 visible characters. Only letters, digits, hyphens (-), and underscores (_) are allowed.                                                                            |
| Version     | Model version. The default value is <b>0.0.1</b> for the first import.<br><b>NOTE</b><br>After a model is created, you can <a href="#">create new versions</a> using different meta models for optimization. |
| Description | Brief description of the model.                                                                                                                                                                              |

- b. Set **Meta Model Source** to **Training job**. For details, see [Table 8-4](#).

**Figure 8-3** Importing a meta model from a training job

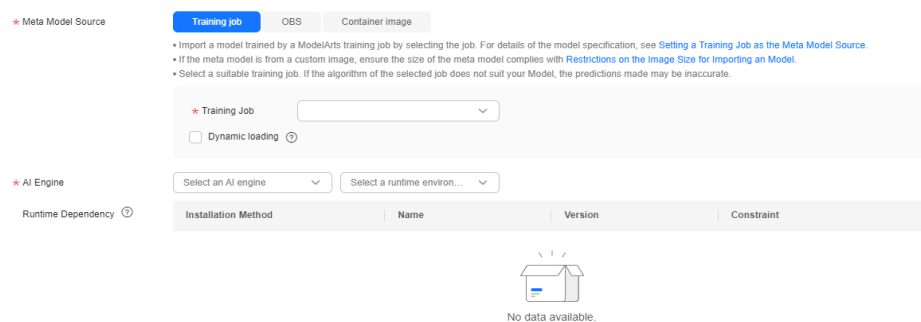


**Table 8-4** Meta model source parameters

| Parameter          | Description                                                                                                                                                                                                                                                                                                                                                                                                    |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Meta Model Source  | Select <b>Training job</b> . <ul style="list-style-type: none"><li>Choose a training job from the <b>Training Job</b> drop-down list.</li><li><b>Dynamic loading</b>: You can enable it for quick model deployment and update. When it is enabled, model files and runtime dependencies are only pulled during an actual deployment. Enable this feature if a single model file is larger than 5 GB.</li></ul> |
| AI Engine          | Inference engine used by the meta model, which is automatically matched based on the training job you select.                                                                                                                                                                                                                                                                                                  |
| Inference Code     | Inference code customizing the inference logic of the model. You can directly copy the inference code URL for use.                                                                                                                                                                                                                                                                                             |
| Runtime Dependency | Dependencies that the selected model has on the environment. For example, if you need to install <b>tensorflow</b> using <b>pip</b> , make sure the version is 1.8.0 or newer.                                                                                                                                                                                                                                 |
| Model Description  | Model descriptions to help other developers better understand and use your model. Click <b>Add Model Description</b> and set the document name and URL. You can add up to three model descriptions.                                                                                                                                                                                                            |
| Deployment Type    | Choose the service types for model deployment. The service types you select will be the only options available for deployment. For example, selecting <b>Real-Time Services</b> means the model can only be deployed as real-time services.                                                                                                                                                                    |

- c. Confirm the configurations and click **Create now**.

In the model list, you can view the created model and its version. When the status changes to **Normal**, the model is created. On this page, you can perform such operations as creating new versions and quickly deploying services.

## Follow-Up Operations

**Deploying a service**: In the model list, click **Deploy** in the **Operation** column of the target model. Locate the target version, click **Deploy** and choose a service type selected during model creation.

### 8.2.3 Importing a Meta Model from OBS

Import a model trained using a mainstream AI engine to ModelArts to create a model for centralized management.

## Constraints

- The imported model, inference code, and configuration file must comply with ModelArts model package specifications. For details, see [Model Package Structure](#), [Specifications for Editing a Model Configuration File](#), and [Specifications for Writing a Model Inference Code File](#).
- If the meta model is from a container image, ensure the size of the meta model complies with [Restrictions on the Size of an Image for Importing a Model](#).
- If you use custom policies as an IAM user and do not use OBS Administrator policy, add these actions:
  - PutObject (uploading using PUT and POST, uploading parts, and initializing and merging uploaded parts)
  - GetObject (obtaining object content and metadata)
  - GetObjectVersion (obtaining the content and metadata of a specified object version)
  - GetObjectAcl (obtaining object ACL)
  - PutObjectAcl (configuring the object ACL)
  - ListBucket (listing objects in the bucket and obtaining the bucket metadata)

## Prerequisites

- The trained model uses an AI engine supported by ModelArts. For details, see [Supported AI Engines for ModelArts Inference](#).
- The model package, inference code, and configuration file have been uploaded to OBS.
- The OBS directory you use and ModelArts are in the same region.

## Procedure

1. Log in to the ModelArts console and choose **Model Management** in the navigation pane on the left.
2. Click **Create Model**.
3. Configure parameters.
  - a. Set basic information about the model. For details about the parameters, see [Table 8-5](#).

**Table 8-5** Basic information

| Parameter | Description                                                                                                                       |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------|
| Name      | Model name. The value can contain 1 to 64 visible characters. Only letters, digits, hyphens (-), and underscores (_) are allowed. |

| Parameter   | Description                                                                                                                                                                                                  |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Version     | Model version. The default value is <b>0.0.1</b> for the first import.<br><b>NOTE</b><br>After a model is created, you can <a href="#">create new versions</a> using different meta models for optimization. |
| Description | Brief description of the model.                                                                                                                                                                              |

- b. Set **Meta Model Source** to **OBS**. For details, see [Table 8-6](#).

To import a meta model from OBS, edit the inference code and configuration file by following [model package specifications](#) and place the inference code and configuration file in the **model** folder storing the meta model. If the selected directory does not comply with the model package specifications, the model cannot be created.

**Figure 8-4** Importing a meta model from OBS

★ Meta Model Source

Training job **OBS** Container image

- Import one of the following models from OBS: TensorFlow, PyTorch, MindSpore, or Custom. To import a model image, you are advised to select Container image. Ensure that the model file is stored in the model directory and specify the parent directory of the model directory as the path. If the model requires inference code, ensure that the code is stored in the model directory. The file name must be "custom\_infer\_service.py". Ensure that the model meets the [Model Package Specifications and Parameters](#).
- When uploading a model larger than 5 GB, select Dynamic loading. It is good practice to set AI Engine to Custom and enable health check for model uploading. [Operation Example](#).
- If the meta model is from a custom image, ensure the size of the meta model complies with [Restrictions on the Image Size for Importing an Model](#).

★ Meta Model

★ AI Engine PyTorch python3.6

Relative Path: /home/ma-user/infer/model/

☐ Dynamic loading ⓘ

Runtime Dependency ⓘ

| Installation Method | Name | Version | Constraint | Operation |
|---------------------|------|---------|------------|-----------|
| <div>Add</div>      |      |         |            |           |

Model Description ⓘ [Add Model Description](#)

Configuration File ☐

**Table 8-6** Meta model source parameters

| Parameter         | Description                                                                                                                                                                                                                    |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Meta Model Source | Select <b>OBS</b> .                                                                                                                                                                                                            |
| Meta Model        | OBS path for storing the meta model.<br>The OBS path cannot contain spaces. Otherwise, the model creation will fail.                                                                                                           |
| AI Engine         | AI engine, which is automatically set according to the model storage path you select, used by the meta model.                                                                                                                  |
| Container API     | This parameter is displayed when <b>AI Engine</b> is set to <b>Custom</b> .<br>Set the protocol and port number of the inference API defined by the model. The default values are <b>HTTPS</b> and <b>8080</b> , respectively. |

| Parameter    | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Health Check | <p>Specifies health check on a model. This parameter is displayed when <b>AI Engine</b> is set to <b>Custom</b>. Once you select a non-custom engine and runtime environment, this parameter is displayed if this engine supports health check.</p> <p>Once you select a custom engine, you must select a container image for the engine package. The health check can be set up only if the container image includes a health check API. Otherwise, the model creation will fail.</p> <p>The following probes are supported:</p> <ul style="list-style-type: none"><li>• <b>Startup Probe:</b> This probe checks if the application instance has started. If a startup probe is provided, all other probes are disabled until it succeeds. If the startup probe fails, the instance is restarted. If no startup probe is provided, the default status is <b>Success</b>.</li><li>• <b>Readiness Probe:</b> This probe verifies whether the application instance is ready to handle traffic. If the readiness probe fails (meaning the instance is not ready), the instance is taken out of the service load balancing pool. Traffic will not be routed to the instance until the probe succeeds.</li><li>• <b>Liveness Probe:</b> This probe monitors the application health status. If the liveness probe fails (indicating the application is unhealthy), the instance is automatically restarted.</li></ul> <p>The parameters of the three types of probes are as follows:</p> <ul style="list-style-type: none"><li>• <b>Check Mode:</b> Retain the default setting <b>HTTP request</b>.</li><li>• <b>Health Check URL:</b> Retain the default setting <b>/health</b>.</li><li>• <b>Health Check Period (s):</b> Enter an integer ranging from 1 to 2147483647.</li><li>• <b>Delay (s):</b> Set a delay for the health check to occur after the instance has started. The value should be an integer between 0 and 2147483647.</li><li>• <b>Timeout (s):</b> Set the timeout interval for each health check. The value should be an integer between 0 and 2147483647.</li><li>• <b>Maximum Failures:</b> Enter an integer ranging from 1 to 2147483647. If the service fails the specified number of consecutive health checks during startup, it will enter the abnormal state. If the service fails the specified number of consecutive health checks during operation, it will enter the alarm state.</li></ul> |



| Parameter          | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                    | <p><b>NOTE</b></p> <p>To use a custom engine to create a model, ensure that the custom engine complies with the specifications for custom engines. For details, see <a href="#">Creating an AI Application Using a Custom Engine</a>.</p> <p>If health check is enabled for a model, the associated services will stop three minutes after receiving the stop instruction.</p>                                                                                                                     |
| Dynamic loading    | You can enable it for quick model deployment and update. When it is enabled, model files and runtime dependencies are only pulled during an actual deployment. Enable this feature if a single model file is larger than 5 GB.                                                                                                                                                                                                                                                                     |
| Runtime Dependency | Dependencies that the selected model has on the environment. For example, if you need to install <b>tensorflow</b> using <b>pip</b> , make sure the version is 1.8.0 or newer.                                                                                                                                                                                                                                                                                                                     |
| Model Description  | Model descriptions to help other developers better understand and use your model. Click <b>Add Model Description</b> and set the document name and URL. You can add up to three descriptions.                                                                                                                                                                                                                                                                                                      |
| Configuration File | <p>The system associates the configuration file stored in OBS by default. After enabling this feature, you can review and edit the model configuration file.</p> <p><b>NOTE</b></p> <p>This feature is to be discontinued. After that, you can modify the model configuration by setting <b>AI Engine</b>, <b>Runtime Dependency</b>, and <b>API Configuration</b>.</p>                                                                                                                            |
| Deployment Type    | Choose the service types for model deployment. The service types you select will be the only options available for deployment. For example, selecting <b>Real-Time Services</b> means the model can only be deployed as real-time services.                                                                                                                                                                                                                                                        |
| Start Command      | <p>Customizable start command of the model. This parameter is optional.</p> <p>When using a preset AI engine, the default start command is used if no start command is entered. For details, see <a href="#">Table 8-2</a>. If a start command is entered, it replaces the default command.</p> <p><b>NOTE</b></p> <p>Start commands containing \$,  , &gt;, &lt;, `, !, \n, \, ?, -v, --volume, --mount, --tmpfs, --privileged, or --cap-add will be emptied when a model is being published.</p> |

| Parameter         | Description                                                                                                                                                                                                                                                                                                                                                                                     |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| API Configuration | After enabling this feature, you can edit RESTful APIs to define the input and output formats of a model. The model APIs must comply with ModelArts specifications. For details, see the <b>apis</b> parameter description in <a href="#">Specifications for Editing a Model Configuration File</a> . For details about the code example, see <a href="#">Code Example of apis Parameters</a> . |

- c. Confirm the configurations and click **Create now**.

In the model list, you can view the created model and its version. When the status changes to **Normal**, the model is created. On this page, you can perform such operations as creating new versions and quickly deploying services.

## Follow-Up Operations

**Deploying a service:** In the model list, click **Deploy** in the **Operation** column of the target model. Locate the target version, click **Deploy** and choose a service type selected during model creation.

## 8.2.4 Importing a Meta Model from a Container Image

For AI engines that are not supported by ModelArts, you can import the models from custom images.

### Constraints

- For details about the specifications and description of custom images, see [Specifications for Custom Images Used for Importing Models](#).
- If the meta model is from a container image, ensure the size of the meta model complies with [restrictions on the size of an image for importing a model](#).
- You must update the custom image version or enable image replication when creating an AI application. Otherwise, new applications will still use the old image version.

### Prerequisites

The OBS directory you use and ModelArts are in the same region.

### Procedure

- Log in to the ModelArts console and choose **Model Management** in the navigation pane on the left.
- Click **Create Model**.
- Configure parameters.
  - Set basic information about the model. For details about the parameters, see [Table 8-7](#).

**Table 8-7** Basic information

| Parameter   | Description                                                                                                                                                                                                  |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Name        | Model name. The value can contain 1 to 64 visible characters. Only letters, digits, hyphens (-), and underscores (_) are allowed.                                                                            |
| Version     | Model version. The default value is <b>0.0.1</b> for the first import.<br><b>NOTE</b><br>After a model is created, you can <a href="#">create new versions</a> using different meta models for optimization. |
| Description | Brief description of the model.                                                                                                                                                                              |

- b. Select the meta model source and configure related parameters. Set **Meta Model Source** to **Container image**. For details, see [Table 8-8](#).

**Figure 8-5** Importing a meta model from a container image

★ Meta Model Source

Training Job OBS **Container image**

• A model imported from a container image is of the image type. Ensure the image can be properly started and provides inference APIs. During service deployment, ModelArts uses the image to deploy inference services. For details about image specifications, see [Learn more](#). Upload an image through a [client](#) or [browser](#). [Parameters](#).

• If the meta model is from a custom image, ensure the size of the meta model complies with [Restrictions on the Image Size for Importing an Model](#).

★ Container Image Path

★ Container API  (host):  Port Number

Image Replication ☐   
When this function is disabled, Model can be created quickly, but modifying or deleting images in the source directory may affect service deployment. When this function is enabled, Model cannot be created quickly, but you can modify or delete images in the source directory as that would not affect service deployment.

Health Check   
Start Probe ☐   
Ready Probe ☐   
Liveness Probe ☐

Model Description [Add Model Description](#)

**Table 8-8** Meta model source parameters

| Parameter            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Container Image Path | Click  to import the container image. You do not need to use <b>swr_location</b> in the configuration file to specify the image location.<br>For details about how to create a custom image, see <a href="#">Specifications for Custom Images Used for Importing Models</a> .<br><b>NOTE</b><br>The model image you select will be shared with the system administrator, so ensure you have the permission to share the image (images shared by other accounts are not supported). ModelArts will deploy the image as an inference service. Ensure that your image can be properly started and provide an inference API. |
| Container API        | Set the protocol and port number of the inference API defined by the model.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

| Parameter         | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Image Replication | <p>Indicates whether to copy the model image in the container image to ModelArts.</p> <ul style="list-style-type: none"><li>• After this feature is disabled, the model image is not copied, models can be rapidly created, but modifying or deleting an image in the SWR source directory will affect service deployment.</li><li>• After this feature is enabled, the model image is copied, models cannot be rapidly created, and modifying or deleting an image in the SWR source directory will not affect service deployment.</li></ul> <p><b>NOTE</b><br/>You must enable this feature if you want to use images shared by others. Otherwise, models will fail to be created.</p> |

| Parameter    | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Health Check | <p>Specifies health check on a model. This parameter is configurable only when a health check API is configured in the custom image. Otherwise, the model creation will fail. The following probes are supported:</p> <ul style="list-style-type: none"> <li>• <b>Startup Probe:</b> This probe checks if the application instance has started. If a startup probe is provided, all other probes are disabled until it succeeds. If the startup probe fails, the instance is restarted. If no startup probe is provided, the default status is <b>Success</b>.</li> <li>• <b>Readiness Probe:</b> This probe verifies whether the application instance is ready to handle traffic. If the readiness probe fails (meaning the instance is not ready), the instance is taken out of the service load balancing pool. Traffic will not be routed to the instance until the probe succeeds.</li> <li>• <b>Liveness Probe:</b> This probe monitors the application health status. If the liveness probe fails (indicating the application is unhealthy), the instance is automatically restarted.</li> </ul> <p>The parameters of the three types of probes are as follows:</p> <ul style="list-style-type: none"> <li>• <b>Check Mode:</b> Select <b>HTTP request</b> or <b>Command</b>.</li> <li>• <b>Health Check URL:</b> Enter the health check URL, which defaults to <b>/health</b>. This parameter is displayed when <b>Check Mode</b> is set to <b>HTTP request</b>.</li> <li>• <b>Health Check Command:</b> Enter the health check command. This parameter is displayed when <b>Check Mode</b> is set to <b>Command</b>.</li> <li>• <b>Health Check Period (s):</b> Enter an integer ranging from 1 to 2147483647.</li> <li>• <b>Delay (s):</b> Set a delay for the health check to occur after the instance has started. The value should be an integer between 0 and 2147483647.</li> <li>• <b>Timeout (s):</b> Set the timeout interval for each health check. The value should be an integer between 0 and 2147483647.</li> <li>• <b>Maximum Failures:</b> Enter an integer ranging from 1 to 2147483647. If the service fails the specified number of consecutive health checks during startup, it will enter the abnormal state. If the service fails the specified number of</li> </ul> |

| Parameter         | Description                                                                                                                                                                                                                                                                                                                                                                                     |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                   | consecutive health checks during operation, it will enter the alarm state.<br><b>NOTE</b><br>If health check is enabled for a model, the associated services will stop three minutes after receiving the stop instruction.                                                                                                                                                                      |
| Model Description | Model descriptions to help other developers better understand and use your model. Click <b>Add Model Description</b> and set the document name and URL. You can add up to three descriptions.                                                                                                                                                                                                   |
| Deployment Type   | Choose the service types for model deployment. The service types you select will be the only options available for deployment. For example, selecting <b>Real-Time Services</b> means the model can only be deployed as real-time services.                                                                                                                                                     |
| Start Command     | Customizable start command of a model.<br><b>NOTE</b><br>Start commands containing \$,  , >, <, `, !, \n, \, ?, -v, --volume, --mount, --tmpfs, --privileged, or --cap-add will be emptied when a model is being published.                                                                                                                                                                     |
| API Configuration | After enabling this feature, you can edit RESTful APIs to define the input and output formats of a model. The model APIs must comply with ModelArts specifications. For details, see the <b>apis</b> parameter description in <a href="#">Specifications for Editing a Model Configuration File</a> . For details about the code example, see <a href="#">Code Example of apis Parameters</a> . |

- c. Confirm the configurations and click **Create now**.

In the model list, you can view the created model and its version. When the status changes to **Normal**, the model is created. On this page, you can perform such operations as creating new versions and quickly deploying services.

## Follow-Up Operations

**Deploying a service:** In the model list, click **Deploy** in the **Operation** column of the target model. Locate the target version, click **Deploy** and choose a service type selected during model creation.

## 8.3 Model Creation Specifications

## 8.3.1 Model Package Structure

When creating a model, make sure that any meta model imported from OBS complies with specifications.

### NOTE

- The model package specifications apply when you import one model. For multiple models, such as multiple files, use custom images instead.
- If you want to use an AI engine that is not supported by ModelArts, use a custom image.
- For details about how to create a custom image, see [Specifications for Custom Images](#) and [Creating a Custom Image on ECS](#).
- For more examples of custom scripts, see [Examples of Custom Scripts](#).

The model package must contain the **model** directory. The **model** directory stores the model file, model configuration file, and model inference code file.

- Model files: The requirements for model files vary according to the model package structure. For details, see [Model Package Example](#).
- Model configuration file: The model configuration file must be available and its name is consistently to be **config.json**. There must be only one model configuration file. For details about how to edit a model configuration file, see [Specifications for Editing a Model Configuration File](#).
- Model inference code file: It is mandatory. The file name is consistently to be **customize\_service.py**. There must be only one model inference code file. For details about how to edit model inference code, see [Specifications for Writing a Model Inference Code File](#).
  - The .py file on which **customize\_service.py** depends can be directly stored in the **model** directory. Use relative import for the custom package.
  - The other files on which **customize\_service.py** depends can be stored in the **model** directory. Use absolute paths to access these files. For details, see [Obtaining an Absolute Path](#).

ModelArts provides samples and sample code for multiple engines. You can edit your configuration files and inference code by referring to [ModelArts Samples](#). ModelArts also provides custom script examples of common AI engines. For details, see [Examples of Custom Scripts](#).

If you encounter any problem when importing a meta model, [contact Huawei Cloud technical support](#).

## Model Package Example

- Structure of the TensorFlow-based model package

When publishing the model, you only need to specify the **ocr** directory.

OBS bucket or directory name

```
|— ocr
| |— model (Mandatory) Name of a fixed subdirectory, which is used to store model-related files
| | |— <<Custom Python package>> (Optional) Your Python package, which can be directly
| | referenced in model inference code
| | |— saved_model.pb (Mandatory) Protocol buffer file, which contains the graph description of
| | the model
| | |— variables Name of a fixed sub-directory, which contains the weight and deviation rate of
```

the model. It is mandatory for the main file of a \*.pb model.

```
| | | | variables.index Mandatory
| | | | variables.data-00000-of-00001 Mandatory
| | | | config.json (Mandatory) Model configuration file. The file name is fixed to config.json.
| | | | Only one model configuration file is allowed.
| | | | customize_service.py (Mandatory) Model inference code. The file name is fixed to
| | | | customize_service.py. Only one model inference code file is allowed.
| | | | The files on which customize_service.py depends can be directly stored in the model directory.
```

- Structure of the PyTorch-based model package

When publishing the model, you only need to specify the **resnet** directory.

OBS bucket or directory name

```
| | resnet
| | | | model (Mandatory) Name of a fixed subdirectory, which is used to store model-related files
| | | | <<Custom Python package>> (Optional) Your Python package, which can be directly
| | | | referenced in model inference code
| | | | resnet50.pth (Mandatory) PyTorch model file, which contains variable and weight
| | | | information and is saved as state_dict
| | | | config.json (Mandatory) Model configuration file. The file name is fixed to config.json.
| | | | Only one model configuration file is allowed.
| | | | customize_service.py (Mandatory) Model inference code. The file name is fixed to
| | | | customize_service.py. Only one model inference code file is allowed. The files on which
| | | | customize_service.py depends can be directly stored in the model directory.
```

- Structure of a custom model package depends on the AI engine in your custom image. For example, if the AI engine in your custom image is TensorFlow, the model package uses the TensorFlow structure.

## 8.3.2 Specifications for Editing a Model Configuration File

You must edit the **config.json** file when publishing a model. The configuration file describes the model usage, computing framework, accuracy, inference code dependency package, and model API.

### Configuration File Format

The configuration file is in JSON format. [Table 8-9](#) describes the parameters.

**Table 8-9** Parameters

| Parameter       | Mandatory | Type   | Description                                                                                                                                                                                                                                                                                                        |
|-----------------|-----------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| model_algorithm | Yes       | String | Model algorithm, which shows the model usage. The value must start with a letter and contain no more than 36 characters. Chinese characters and special characters (&!"'<>=) are not allowed. Major model algorithms include <b>image_classification</b> , <b>object_detection</b> , and <b>predict_analysis</b> . |



| Parameter  | Mandatory | Type   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|------------|-----------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| model_type | Yes       | String | <p>Model AI engine, which indicates the computing framework used by a model. Major AI engines and <b>Image</b> are supported.</p> <ul style="list-style-type: none"><li>For details about supported AI engines, see <a href="#">Supported AI Engines for ModelArts Inference</a>.</li><li>If <b>model_type</b> is set to <b>Image</b>, a model will be created using a custom image. In this case, the <b>swr_location</b> parameter is mandatory. For details about how to create an image, see <a href="#">Specifications for Custom Images</a>.</li></ul>                                                                                                                                                    |
| runtime    | No        | String | <p>Model runtime environment. Python 2.7 is used by default. The value of <b>runtime</b> depends on <b>model_type</b>. If <b>model_type</b> is set to <b>Image</b>, you do not need to set <b>runtime</b>. If <b>model_type</b> is set to a mainstream framework, select a runtime environment matching the engine. For details about the supported runtime environments, see <a href="#">Supported AI Engines for ModelArts Inference</a>.</p> <p>If your model must run on specified CPUs or GPUs, select the CPUs or GPUs based on the runtime suffix. If the runtime does not contain the CPU or GPU information, check the runtime description in <i>Supported AI Engines for ModelArts Inference</i>.</p> |
| metrics    | No        | Object | <p>Model precision information, including the F1 score, recall, precision, and accuracy. For details about the <b>metrics</b> object structure, see <a href="#">Table 8-10</a>.</p> <p>The result is displayed in the model precision area on the model details page.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                       |

| Parameter    | Mandatory | Type                  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|--------------|-----------|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| apis         | No        | API array             | <p>Structure data of requests received and returned by a model.</p> <p>It is a RESTful API array provided by a model. For details about the API structure, see <a href="#">Table 8-11</a>. For details about the code example, see <a href="#">Code Example of apis Parameters</a>.</p> <ul style="list-style-type: none"><li>• If <b>model_type</b> is set to <b>Image</b>, an AI application will be created using a custom image. APIs with different paths can be declared in <b>apis</b> based on the request path exposed by the image.</li><li>• If <b>model_type</b> is not <b>Image</b>, only one API whose request path is / can be declared in <b>apis</b> because the preconfigured AI engine exposes only one inference API whose request path is /.</li></ul>                                                                                                                                                                                                                                                                      |
| dependencies | No        | Dependency array      | <p>Package on which the model inference code depends, which is structure data.</p> <p>You must provide the package name, installation method, and version constraints. The dependency package can be installed only using <b>pip</b>. <a href="#">Table 8-14</a> describes the dependency array.</p> <p>If the model package does not contain the <b>customize_service.py</b> inference code file, you do not need to set dependencies. Dependency packages cannot be installed for custom image models.</p> <p><b>NOTE</b></p> <p>The <b>dependencies</b> parameter accepts multiple dependency arrays in a list format. It applies to scenarios where the default installation packages have dependency relationships. The top packages are installed first. The wheel package can be used for dependency installation, and it must be stored in the same directory as the model file. For details, see <a href="#">How Do I Edit the Installation Package Dependency Parameters in a Model Configuration File When Importing a Model?</a></p> |
| health       | No        | health data structure | <p>Health check API configuration. This parameter is mandatory only when <b>model_type</b> is set to <b>Image</b>.</p> <p>To ensure uninterrupted services during a rolling upgrade, ModelArts requires a health check API. For details about the <b>health</b> data structure, see <a href="#">Table 8-16</a>.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

Table 8-10 metrics object

| Parameter | Mandatory | Type   | Description                                           |
|-----------|-----------|--------|-------------------------------------------------------|
| f1        | No        | Number | F1 score. The value is rounded to 17 decimal places.  |
| recall    | No        | Number | Recall. The value is rounded to 17 decimal places.    |
| precision | No        | Number | Precision. The value is rounded to 17 decimal places. |
| accuracy  | No        | Number | Accuracy. The value is rounded to 17 decimal places.  |

Table 8-11 api structure

| Parameter | Mandatory | Type   | Description                                                                                                                                                                                                                                                                        |
|-----------|-----------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| url       | No        | String | Request path. The default value is a slash (/). For a custom image model ( <b>model_type</b> is <b>Image</b> ), set this parameter to the actual request path exposed in the image. For a non-custom image model ( <b>model_type</b> is not <b>Image</b> ), the URL can only be /. |
| method    | No        | String | Request method. The default value is <b>POST</b> .                                                                                                                                                                                                                                 |
| request   | No        | Object | Request body. For details, see <a href="#">Table 8-12</a> .                                                                                                                                                                                                                        |
| response  | No        | Object | Response body. For details, see <a href="#">Table 8-13</a> .                                                                                                                                                                                                                       |

Table 8-12 request structure

| Parameter    | Mandatory                                           | Type   | Description                                                                                                                                                                                                                                                                                                                                                                  |
|--------------|-----------------------------------------------------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Content-type | No for real-time services<br>Yes for batch services | String | Data is sent in a specified content format. The default value is <b>application/json</b> .<br>The options are as follows: <ul style="list-style-type: none"><li>• <b>application/json</b>: JSON data is uploaded.</li><li>• <b>multipart/form-data</b>: A file is uploaded.</li></ul> <b>NOTE</b><br>For machine learning models, only <b>application/json</b> is supported. |

| Parameter | Mandatory                                           | Type   | Description                                                                                                                         |
|-----------|-----------------------------------------------------|--------|-------------------------------------------------------------------------------------------------------------------------------------|
| data      | No for real-time services<br>Yes for batch services | String | The request body is described in JSON schema. For details about the parameter description, see the <a href="#">official guide</a> . |

**Table 8-13 response structure**

| Parameter    | Mandatory                                           | Type   | Description                                                                                                                                                                          |
|--------------|-----------------------------------------------------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Content-type | No for real-time services<br>Yes for batch services | String | Data is sent in a specified content format. The default value is <b>application/json</b> .<br><b>NOTE</b><br>For machine learning models, only <b>application/json</b> is supported. |
| data         | No for real-time services<br>Yes for batch services | String | The response body is described in JSON schema. For details about the parameter description, see the <a href="#">official guide</a> .                                                 |

**Table 8-14 dependency array**

| Parameter | Mandatory | Type          | Description                                                                                          |
|-----------|-----------|---------------|------------------------------------------------------------------------------------------------------|
| installer | Yes       | String        | Installation method. Only <b>pip</b> is supported.                                                   |
| packages  | Yes       | package array | Dependency package collection. For details about the package array, see <a href="#">Table 8-15</a> . |

**Table 8-15** Package array

| Parameter       | Mandatory | Type   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-----------------|-----------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| package_name    | Yes       | String | Dependency package name. Chinese characters and special characters (&!"'<>=) are not allowed.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| package_version | No        | String | Dependency package version. Leave it blank if it is not required. Chinese characters and special characters (&!"'<>=) are not allowed.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| restraint       | No        | String | <p>Version restriction. This parameter is mandatory only when <b>package_version</b> is configured. Possible values are <b>EXACT</b>, <b>ATLEAST</b>, and <b>ATMOST</b>.</p> <ul style="list-style-type: none"><li>• <b>EXACT</b> indicates that a specified version is installed.</li><li>• <b>ATLEAST</b> indicates that the installed version is not earlier than the specified version.</li><li>• <b>ATMOST</b> indicates that the installed version is not later than the specified version.</li></ul> <p><b>NOTE</b></p> <ul style="list-style-type: none"><li>• If there are specific requirements on the version, preferentially use <b>EXACT</b>. If <b>EXACT</b> conflicts with the system installation packages, you can select <b>ATLEAST</b>.</li><li>• If there is no specific requirement on the version, retain only the <b>package_name</b> parameter and leave <b>restraint</b> and <b>package_version</b> blank.</li></ul> |

**Table 8-16** Health check data structure

| Parameter    | Mandatory | Type   | Description                                                                                                                                                                                            |
|--------------|-----------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| check_method | Yes       | String | <p>Health check method. The value can be <b>HTTP</b> or <b>EXEC</b>.</p> <ul style="list-style-type: none"><li>• <b>HTTP</b>: Use an HTTP request.</li><li>• <b>EXEC</b>: Execute a command.</li></ul> |

| Parameter             | Mandatory | Type   | Description                                                                                                                                             |
|-----------------------|-----------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| command               | No        | String | Health check command. This parameter is mandatory when <b>check_method</b> is set to <b>EXEC</b> .                                                      |
| url                   | No        | String | Request URL of a health check API. This parameter is mandatory when <b>check_method</b> is set to <b>HTTP</b> .                                         |
| protocol              | No        | String | Request protocol of a health check API. The default value is <b>http</b> . This parameter is mandatory when <b>check_method</b> is set to <b>HTTP</b> . |
| initial_delay_seconds | No        | String | Delay for initializing the health check.                                                                                                                |
| timeout_seconds       | No        | String | Health check timeout.                                                                                                                                   |
| period_seconds        | Yes       | String | Health check period, in seconds. Enter a positive integer no more than 2147483647.                                                                      |
| failure_threshold     | Yes       | String | Maximum number of health check failures. Enter a positive integer no more than 2147483647.                                                              |

## Code Example of apis Parameters

```
[{
 "url": "/",
 "method": "post",
 "request": {
 "Content-type": "multipart/form-data",
 "data": {
 "type": "object",
 "properties": {
 "images": {
 "type": "file"
 }
 }
 }
 },
 "response": {
 "Content-type": "application/json",
 "data": {
 "type": "object",
 "properties": {
 "mnist_result": {
 "type": "array",
 "item": [
 {
 "type": "string"
 }
]
 }
 }
 }
 }
}]
```

```
}
}
}
}
}]
```

## Example of an Object Detection Model Configuration File

The following code uses the TensorFlow engine as an example. You can modify the **model\_type** parameter based on the actual engine type.

- Model input

Key: images

Value: image files

- Model output

```
{
 "detection_classes": [
 "face",
 "arm"
],
 "detection_boxes": [
 [
 33.6,
 42.6,
 104.5,
 203.4
],
 [
 103.1,
 92.8,
 765.6,
 945.7
]
],
 "detection_scores": [0.99, 0.73]
}
```

- Configuration file

```
{
 "model_type": "TensorFlow",
 "model_algorithm": "object_detection",
 "metrics": {
 "f1": 0.345294,
 "accuracy": 0.462963,
 "precision": 0.338977,
 "recall": 0.351852
 },
 "apis": [{
 "url": "/",
 "method": "post",
 "request": {
 "Content-type": "multipart/form-data",
 "data": {
 "type": "object",
 "properties": {
 "images": {
 "type": "file"
 }
 }
 }
 }
 }
},
 "response": {
 "Content-type": "application/json",
 "data": {
 "type": "object",
 "properties": {
 "detection_classes": [
 "face",
 "arm"
],
 "detection_boxes": [
 [
 33.6,
 42.6,
 104.5,
 203.4
],
 [
 103.1,
 92.8,
 765.6,
 945.7
]
],
 "detection_scores": [0.99, 0.73]
 }
 }
 }
}
```

```
 "detection_classes": {
 "type": "array",
 "items": [{
 "type": "string"
 }]
 },
 "detection_boxes": {
 "type": "array",
 "items": [{
 "type": "array",
 "minItems": 4,
 "maxItems": 4,
 "items": [{
 "type": "number"
 }]
 }]
 },
 "detection_scores": {
 "type": "array",
 "items": [{
 "type": "number"
 }]
 }
 }
 }
 },
 "dependencies": [{
 "installer": "pip",
 "packages": [{
 "restraint": "EXACT",
 "package_version": "1.15.0",
 "package_name": "numpy"
 },
 {
 "restraint": "EXACT",
 "package_version": "5.2.0",
 "package_name": "Pillow"
 }
]
}]
}
```

## Example of an Image Classification Model Configuration File

The following code uses the TensorFlow engine as an example. You can modify the **model\_type** parameter based on the actual engine type.

- Model input

Key: images

Value: image files

- Model output

```
{
 "predicted_label": "flower",
 "scores": [
 ["rose", 0.99],
 ["begonia", 0.01]
]
}
```

- Configuration file

```
{
 "model_type": "TensorFlow",
 "model_algorithm": "image_classification",
 "metrics": {
 "f1": 0.345294,
```



```
 "accuracy": 0.462963,
 "precision": 0.338977,
 "recall": 0.351852
 },
 "apis": [{
 "url": "/",
 "method": "post",
 "request": {
 "Content-type": "multipart/form-data",
 "data": {
 "type": "object",
 "properties": {
 "images": {
 "type": "file"
 }
 }
 }
 }
 }],
 "response": {
 "Content-type": "application/json",
 "data": {
 "type": "object",
 "properties": {
 "predicted_label": {
 "type": "string"
 },
 "scores": {
 "type": "array",
 "items": [{
 "type": "array",
 "minItems": 2,
 "maxItems": 2,
 "items": [
 {
 "type": "string"
 },
 {
 "type": "number"
 }
]
 }
]
 }
 }
 }
}],
 "dependencies": [{
 "installer": "pip",
 "packages": [{
 "restraint": "ATLEAST",
 "package_version": "1.15.0",
 "package_name": "numpy"
 },
 {
 "restraint": "",
 "package_version": "",
 "package_name": "Pillow"
 }
]
}]
}
```

The following code uses the MindSpore engine as an example. You can modify the **model\_type** parameter based on the actual engine type.

- Model input  
Key: images  
Value: image files

- Model output

```
"[[-2.404526 -3.0476532 -1.9888215 0.45013925 -1.7018927 0.40332815\n11.290332 -1.5861531 5.7887416]]"
```

- Configuration file

```
{
 "model_algorithm": "image_classification",
 "model_type": "MindSpore",
 "metrics": {
 "f1": 0.124555,
 "recall": 0.171875,
 "precision": 0.0023493892851938493,
 "accuracy": 0.00746268656716417
 },
 "apis": [{
 "url": "/",
 "method": "post",
 "request": {
 "Content-type": "multipart/form-data",
 "data": {
 "type": "object",
 "properties": {
 "images": {
 "type": "file"
 }
 }
 }
 },
 "response": {
 "Content-type": "application/json",
 "data": {
 "type": "object",
 "properties": {
 "mnist_result": {
 "type": "array",
 "item": [{
 "type": "string"
 }]
 }
 }
 }
 }
]
},
"dependencies": []
}
```

## Example of a Predictive Analytics Model Configuration File

The following code uses the TensorFlow engine as an example. You can modify the **model\_type** parameter based on the actual engine type.

- Model input

```
{
 "data": {
 "req_data": [
 {
 "buying_price": "high",
 "maint_price": "high",
 "doors": "2",
 "persons": "2",
 "lug_boot": "small",
 "safety": "low",
 "acceptability": "acc"
 },
 {
 "buying_price": "high",
 "maint_price": "high",

```

```
 "doors": "2",
 "persons": "2",
 "lug_boot": "small",
 "safety": "low",
 "acceptability": "acc"
 }
]
 }
}
```

- Model output

```
{
 "data": {
 "resp_data": [
 {
 "predict_result": "unacc"
 },
 {
 "predict_result": "unacc"
 }
]
 }
}
```

- Configuration file

 **NOTE**

In the code, the **data** parameter in the request and response structures is described in JSON Schema. The content in **data** and **properties** corresponds to the model input and output.

```
{
 "model_type": "TensorFlow",
 "model_algorithm": "predict_analysis",
 "metrics": {
 "f1": 0.345294,
 "accuracy": 0.462963,
 "precision": 0.338977,
 "recall": 0.351852
 },
 "apis": [
 {
 "url": "/",
 "method": "post",
 "request": {
 "Content-type": "application/json",
 "data": {
 "type": "object",
 "properties": {
 "data": {
 "type": "object",
 "properties": {
 "req_data": {
 "items": [
 {
 "type": "object",
 "properties": {}
 }
],
 "type": "array"
 }
 }
 }
 }
 }
 },
 "response": {
 "Content-type": "application/json",
 "data": {
 "type": "object",

```

```
"properties": {
 "data": {
 "type": "object",
 "properties": {
 "resp_data": {
 "type": "array",
 "items": [
 {
 "type": "object",
 "properties": {}
 }
]
 }
 }
 }
},
"dependencies": [
 {
 "installer": "pip",
 "packages": [
 {
 "restraint": "EXACT",
 "package_version": "1.15.0",
 "package_name": "numpy"
 },
 {
 "restraint": "EXACT",
 "package_version": "5.2.0",
 "package_name": "Pillow"
 }
]
 }
]
```

## Example of a Custom Image Model Configuration File

The model input and output are similar to those in [Example of an Object Detection Model Configuration File](#).

- Here is an example of how to make a model prediction request for image files.

To upload files, click the file upload button on the inference page.

```
{
 "Content-type": "multipart/form-data",
 "data": {
 "type": "object",
 "properties": {
 "images": {
 "type": "file"
 }
 }
 }
}
```

- Here is an example of how to make a model prediction request for JSON data.

The input parameter is of string type. To enter prediction requests, use the text box on the inference page.

```
{
 "Content-type": "application/json",
```

```
 "data": {
 "type": "object",
 "properties": {
 "input": {
 "type": "string"
 }
 }
 }
 }
}
```

A complete request example is as follows:

```
{
 "model_algorithm": "image_classification",
 "model_type": "Image",
 "metrics": {
 "f1": 0.345294,
 "accuracy": 0.462963,
 "precision": 0.338977,
 "recall": 0.351852
 },
 "apis": [{
 "url": "/",
 "method": "post",
 "request": {
 "Content-type": "multipart/form-data",
 "data": {
 "type": "object",
 "properties": {
 "images": {
 "type": "file"
 }
 }
 }
 }
]},
 "response": {
 "Content-type": "application/json",
 "data": {
 "type": "object",
 "required": [
 "predicted_label",
 "scores"
],
 "properties": {
 "predicted_label": {
 "type": "string"
 },
 "scores": {
 "type": "array",
 "items": [{
 "type": "array",
 "minItems": 2,
 "maxItems": 2,
 "items": [{
 "type": "string"
 },
 {
 "type": "number"
 }
]
 }
]
 }
]}
}
```

## Example of a Machine Learning Model Configuration File

The following uses XGBoost as an example:

- Model input

```
{
 "req_data": [
 {
 "sepal_length": 5,
 "sepal_width": 3.3,
 "petal_length": 1.4,
 "petal_width": 0.2
 },
 {
 "sepal_length": 5,
 "sepal_width": 2,
 "petal_length": 3.5,
 "petal_width": 1
 },
 {
 "sepal_length": 6,
 "sepal_width": 2.2,
 "petal_length": 5,
 "petal_width": 1.5
 }
]
}
```

- Model output

```
{
 "resp_data": [
 {
 "predict_result": "Iris-setosa"
 },
 {
 "predict_result": "Iris-versicolor"
 }
]
}
```

- Configuration file

```
{
 "model_type": "XGBoost",
 "model_algorithm": "xgboost_iris_test",
 "runtime": "python2.7",
 "metrics": {
 "f1": 0.345294,
 "accuracy": 0.462963,
 "precision": 0.338977,
 "recall": 0.351852
 },
 "apis": [
 {
 "url": "/",
 "method": "post",
 "request": {
 "Content-type": "application/json",
 "data": {
 "type": "object",
 "properties": {
 "req_data": {
 "items": [
 {
 "type": "object",
 "properties": {}
 }
]
 },
 "type": "array"
 }
 }
 }
 }
]
}
```

```
 }
 }
},
"response": {
 "Content-type": "application/json",
 "data": {
 "type": "object",
 "properties": {
 "resp_data": {
 "type": "array",
 "items": [
 {
 "type": "object",
 "properties": {
 "predict_result": {}
 }
 }
]
 }
 }
 }
}
}
```

## Example of the Model Configuration File Using a Custom Dependency Package

The following example defines the NumPy 1.16.4 dependency environment.

```
{
 "model_algorithm": "image_classification",
 "model_type": "TensorFlow",
 "runtime": "python3.6",
 "apis": [
 {
 "url": "/",
 "method": "post",
 "request": {
 "Content-type": "multipart/form-data",
 "data": {
 "type": "object",
 "properties": {
 "images": {
 "type": "file"
 }
 }
 }
 }
 }
],
 "response": {
 "Content-type": "application/json",
 "data": {
 "type": "object",
 "properties": {
 "mnist_result": {
 "type": "array",
 "item": [
 {
 "type": "string"
 }
]
 }
 }
 }
 }
}
```

```
 },
],
 "metrics": {
 "f1": 0.124555,
 "recall": 0.171875,
 "precision": 0.00234938928519385,
 "accuracy": 0.00746268656716417
 },
 "dependencies": [
 {
 "installer": "pip",
 "packages": [
 {
 "restraint": "EXACT",
 "package_version": "1.16.4",
 "package_name": "numpy"
 }
]
 }
]
}
```

### 8.3.3 Specifications for Writing a Model Inference Code File

This section describes the general method of editing model inference code in ModelArts. For details about the custom script examples (including inference code examples) of mainstream AI engines, see [Examples of Custom Scripts](#). This section also provides an inference code example for the TensorFlow engine and an example of customizing the inference logic in the inference script.

Due to the limitation of API Gateway, the duration of a single prediction in ModelArts cannot exceed 40s. The model inference code must be logically clear and concise for satisfactory inference performance.

#### Specifications for Writing Inference Code

1. In the model inference code file **customize\_service.py**, add a child model class. This child model class inherits properties from its parent model class. For details about the import statements of different types of parent model classes, see [Table 8-17](#). The ModelArts environment has already configured the necessary Python packages for import statements, so you do not need to install them separately.

**Table 8-17** Parent class and import statement of each model type

| Model Type | Parent Class         | Import Statement                                                           |
|------------|----------------------|----------------------------------------------------------------------------|
| TensorFlow | TfServingBaseService | from model_service.tf-serving_model_service<br>import TfServingBaseService |
| PyTorch    | PTServingBaseService | from model_service.pytorch_model_service<br>import PTServingBaseService    |
| MindSpore  | SingleNodeService    | from model_service.model_service import<br>SingleNodeService               |

2. The following methods can be overridden.



**Table 8-18** Methods to be overridden

| Method                                              | Description                                                                                                                                                                                                                                                                     |
|-----------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__init__(self, model_name, model_path)</code> | Initialization method, which is suitable for models created based on deep learning frameworks. Models and labels are loaded using this method. To implement model loading logic, override this method for PyTorch and Caffe-based models.                                       |
| <code>__init__(self, model_path)</code>             | Initialization method, which is suitable for models created based on machine learning frameworks. This method initializes the model path ( <b><code>self.model_path</code></b> ). In Spark_MLLib, this method also initializes SparkSession ( <b><code>self.spark</code></b> ). |
| <code>_preprocess(self, data)</code>                | Preprocess method, which is called before an inference request and converts API request data into the model's expected input format.                                                                                                                                            |
| <code>_inference(self, data)</code>                 | Inference request method. You are advised not to override this method, as it will replace the built-in inference process in ModelArts with your custom logic.                                                                                                                   |
| <code>_postprocess(self, data)</code>               | Postprocess method, which is called after an inference request is complete and converts the model output to the API output.                                                                                                                                                     |

 **NOTE**

- You can override the preprocess and postprocess methods for preprocessing the API input and postprocessing the inference output.
  - Overriding the init method of the parent model class may cause a model to run abnormally.
3. The attribute that can be used is the local path to the model. The attribute name is **`self.model_path`**. Additionally, PySpark-based models can use **`self.spark`** to obtain the SparkSession object in **`customize_service.py`**.

 **NOTE**

The inference code requires an absolute file path for reading files. You can obtain the local path to the model from the **`self.model_path`** attribute.

- When TensorFlow, Caffe, or MXNet is used, **`self.model_path`** indicates the path to the model file. The following provides an example:  

```
Reads the label.json file in the model directory.
with open(os.path.join(self.model_path, 'label.json')) as f:
 self.label = json.load(f)
```
- When PyTorch, Scikit\_Learn, or PySpark is used, **`self.model_path`** indicates the path to the model file. The following provides an example:  

```
Reads the label.json file in the model directory.
dir_path = os.path.dirname(os.path.realpath(self.model_path))
with open(os.path.join(dir_path, 'label.json')) as f:
 self.label = json.load(f)
```

4. The API accepts data in either **`multipart/form-data`** or **`application/json`** format for pre-processing, actual inference, and post-processing.

- **multipart/form-data** request

```
curl -X POST \
<modelarts-inference-endpoint> \
-F image1=@cat.jpg \
-F images2=@horse.jpg
```

The input data is as follows:

```
[
 {
 "image1":{
 "cat.jpg":"<cat.jpg file io>"
 }
 },
 {
 "image2":{
 "horse.jpg":"<horse.jpg file io>"
 }
 }
]
```

- **application/json** request

```
curl -X POST \
<modelarts-inference-endpoint> \
-d '{
 "images":"base64 encode image"
}'
```

The input data is **python dict**.

```
{
 "images":"base64 encode image"
}
```

## TensorFlow Inference Script Example

The following is an example of TensorFlow MnistService. For more TensorFlow inference code examples, see [Tensorflow](#) and [Tensorflow2.1](#).

- Inference code

```
from PIL import Image
import numpy as np
from model_service.tf-serving_model_service import TfServingBaseService

class MnistService(TfServingBaseService):

 def _preprocess(self, data):
 preprocessed_data = {}

 for k, v in data.items():
 for file_name, file_content in v.items():
 image1 = Image.open(file_content)
 image1 = np.array(image1, dtype=np.float32)
 image1.resize((1, 784))
 preprocessed_data[k] = image1

 return preprocessed_data

 def _postprocess(self, data):
 infer_output = {}

 for output_name, result in data.items():
 infer_output["mnist_result"] = result[0].index(max(result[0]))

 return infer_output
```

- Request

```
curl -X POST \ Real-time service address \ -F images=@test.jpg
```

- Response  
`{"mnist_result": 7}`

The preceding sample code resizes images imported to the user's form to adapt to the model input shape. The **32×32** image is read from the Pillow library and resized to **1×784** to match the model input. In subsequent processing, convert the model output into a list for the RESTful API to display.

## Inference Script Example of Custom Inference Logic

Customize a dependency package in the configuration file by referring to [Example of the Model Configuration File Using a Custom Dependency Package](#). Then, use the following code example to load the model in **saved\_model** format for inference.

### NOTE

Python logging used by inference base images allows the display of only warning logs. To query INFO logs, set the log level to INFO in the code.

```
-*- coding: utf-8 -*-
import json
import os
import threading
import numpy as np
import tensorflow as tf
from PIL import Image
from model_service.tf_serving_model_service import TfServingBaseService
import logging
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(name)s - %(levelname)s - %(message)s')
logger = logging.getLogger(__name__)

class MnistService(TfServingBaseService):
 def __init__(self, model_name, model_path):
 self.model_name = model_name
 self.model_path = model_path
 self.model_inputs = {}
 self.model_outputs = {}

 # The label file can be loaded here and used in the post-processing function.
 # Directories for storing the label.txt file on OBS and in the model package

 # with open(os.path.join(self.model_path, 'label.txt')) as f:
 # self.label = json.load(f)

 # Load the model in saved_model format in non-blocking mode to prevent blocking timeout.
 thread = threading.Thread(target=self.get_tf_sess)
 thread.start()

 def get_tf_sess(self):
 # Load the model in saved_model format.
 # The session will be reused. Do not use the with statement.
 sess = tf.Session(graph=tf.Graph())
 meta_graph_def = tf.saved_model.loader.load(sess, [tf.saved_model.tag_constants.SERVING],
self.model_path)
 signature_defs = meta_graph_def.signature_def
 self.sess = sess
 signature = []

 # only one signature allowed
 for signature_def in signature_defs:
 signature.append(signature_def)
 if len(signature) == 1:
 model_signature = signature[0]
 else:
 logger.warning("signatures more than one, use serving_default signature")
```

```
model_signature = tf.saved_model.signature_constants.DEFAULT_SERVING_SIGNATURE_DEF_KEY

logger.info("model signature: %s", model_signature)

for signature_name in meta_graph_def.signature_def[model_signature].inputs:
 tensorinfo = meta_graph_def.signature_def[model_signature].inputs[signature_name]
 name = tensorinfo.name
 op = self.sess.graph.get_tensor_by_name(name)
 self.model_inputs[signature_name] = op

logger.info("model inputs: %s", self.model_inputs)

for signature_name in meta_graph_def.signature_def[model_signature].outputs:
 tensorinfo = meta_graph_def.signature_def[model_signature].outputs[signature_name]
 name = tensorinfo.name
 op = self.sess.graph.get_tensor_by_name(name)
 self.model_outputs[signature_name] = op

logger.info("model outputs: %s", self.model_outputs)

def _preprocess(self, data):
 # Two HTTPS request formats
 # 1. Request in form-data format: data = {"Request key value":{"File name":<File io>}}
 # 2. Request in JSON format: data = json.loads("JSON body passed in the API")
 preprocessed_data = {}

 for k, v in data.items():
 for file_name, file_content in v.items():
 image1 = Image.open(file_content)
 image1 = np.array(image1, dtype=np.float32)
 image1.resize((1, 28, 28))
 preprocessed_data[k] = image1

 return preprocessed_data

def _inference(self, data):
 feed_dict = {}
 for k, v in data.items():
 if k not in self.model_inputs.keys():
 logger.error("input key %s is not in model inputs %s", k, list(self.model_inputs.keys()))
 raise Exception("input key %s is not in model inputs %s" % (k, list(self.model_inputs.keys())))
 feed_dict[self.model_inputs[k]] = v

 result = self.sess.run(self.model_outputs, feed_dict=feed_dict)
 logger.info('predict result : ' + str(result))
 return result

def _postprocess(self, data):
 infer_output = {"mnist_result": []}
 for output_name, results in data.items():

 for result in results:
 infer_output["mnist_result"].append(np.argmax(result))

 return infer_output

def __del__(self):
 self.sess.close()
```

 NOTE

To load multiple models or models that are not supported by ModelArts, specify the loading path using the `__init__` method. Example code:

```
-*- coding: utf-8 -*-
import os
from model_service.tf-serving_model_service import TfServingBaseService

class MnistService(TfServingBaseService):
 def __init__(self, model_name, model_path):
 # Obtain the path to the model folder.
 root = os.path.dirname(os.path.abspath(__file__))
 # test.onnx is the name of the model file to be loaded and must be stored in the model folder.
 self.model_path = os.path.join(root, test.onnx)

 # Load multiple models, for example, test2.onnx.
 # self.model_path2 = os.path.join(root, test2.onnx)
```

### 8.3.4 Specifications for Using a Custom Engine to Create a Model

When using a custom engine to create a model, you can select your image stored in SWR as the engine and specify a file directory in OBS as the model package. In this way, bring-your-own images can be used to meet your dedicated requirements.

Before deploying such a model as a service, ModelArts downloads the SWR image to the cluster and starts the image as a container as the user whose UID is 1000 and GID is 100. Then, ModelArts downloads the OBS file to the **/home/mind/model** directory in the container and runs the boot command preset in the SWR image. ModelArts registers an inference API with API Gateway for you to access the service.

#### Specifications for Using a Custom Engine to Create a Model

To use a custom engine to create a model, ensure the SWR image, OBS model package, and file size comply with the following requirements:

- SWR image specifications
  - A common user named **ma-user** in group **ma-group** must be built in the SWR image. Additionally, the UID and GID of the user must be 1000 and 100, respectively. The following is the dockerfile command for the built-in user:  
groupadd -g 100 ma-group && useradd -d /home/ma-user -m -u 1000 -g 100 -s /bin/bash ma-user
  - Specify a command for starting the image. In the dockerfile, specify **cmd**. The following shows an example:  
CMD sh /home/mind/run.sh

Customize the boot file **run.sh**. The following is an example:

```
#!/bin/bash

User-defined script content
...

run.sh calls app.py to start the server. For details about app.py, see "HTTPS Example".
python app.py
```

 NOTE

You can also customize the boot command for starting an image. Enter the customized command during model creation.

- The provided service can use the HTTPS/HTTP protocol and listening container port. Set the protocol and port number based on the inference API defined by the model. For details about the HTTPS protocol, see [HTTPS Example](#).
- (Optional) On port provided by the service for external access, enable health check with URL **/health**. (The health check URL must be **/health**.)
- OBS model package specifications  
The name of the model package must be **model**. For details about the model package specifications, see [Model Package Specifications](#).
- File size specifications  
When a public resource pool is used, the total size of the downloaded SWR image (not the compressed image displayed on the SWR page) and the OBS model package cannot exceed 30 GB.

## HTTPS Example

Use Flask to start HTTPS. The following is an example of the web server code:

```
from flask import Flask, request
import json

app = Flask(__name__)

@app.route('/greet', methods=['POST'])
def say_hello_func():
 print("----- in hello func -----")
 data = json.loads(request.get_data(as_text=True))
 print(data)
 username = data['name']
 rsp_msg = 'Hello, {}'.format(username)
 return json.dumps({"response":rsp_msg}, indent=4)

@app.route('/goodbye', methods=['GET'])
def say_goodbye_func():
 print("----- in goodbye func -----")
 return '\nGoodbye!\n'

@app.route('/', methods=['POST'])
def default_func():
 print("----- in default func -----")
 data = json.loads(request.get_data(as_text=True))
 return '\n called default func !\n {} \n'.format(str(data))

@app.route('/health', methods=['GET'])
def healthy():
 return "{\"status\": \"OK\"}"

host must be "0.0.0.0", port must be 8080
if __name__ == '__main__':
 app.run(host="0.0.0.0", port=8080, ssl_context='adhoc')
```

## Debugging on a Local Computer

Perform the following operations on a local computer with Docker installed to check whether a custom engine complies with specifications:

1. Download the custom image, for example, **custom\_engine:v1** to the local computer.
2. Copy the model package folder **model** to the local computer.
3. Run the following command in the same directory as the model package folder to start the service:

```
docker run --user 1000:100 -p 8080:8080 -v model:/home/mind/model custom_engine:v1
```

#### NOTE

This command is used for simulation only because the directory mounted to **-v** is assigned the root permission. In the cloud environment, after the model file is downloaded from OBS to **/home/mind/model**, the file owner will be changed to **ma-user**.

4. Start another terminal on the local computer and run the following command to obtain the expected inference result:
- ```
curl https://127.0.0.1:8080/${Request path to the inference service}
```

Deployment Example

The following section describes how to use a custom engine to create a model.

1. Create a model and view model details.

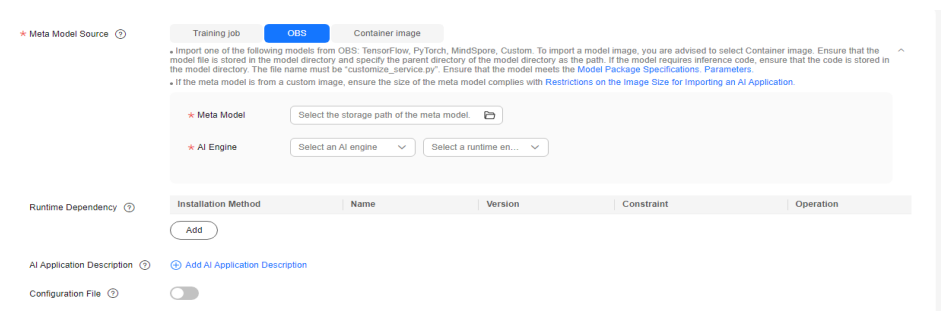
Log in to the ModelArts console, choose **Model Management** from the navigation pane and click **Create Model**. Configure the following parameters:

- **Meta Model Source:** Select **OBS**.
- **Meta Model:** Select a model package from OBS.
- **AI Engine:** Select **Custom**.
- **Engine Package:** Select a container image from SWR.

Retain default settings for other parameters.

Click **Create now**. Wait until the model status changes to **Normal**.

Figure 8-6 Creating a model



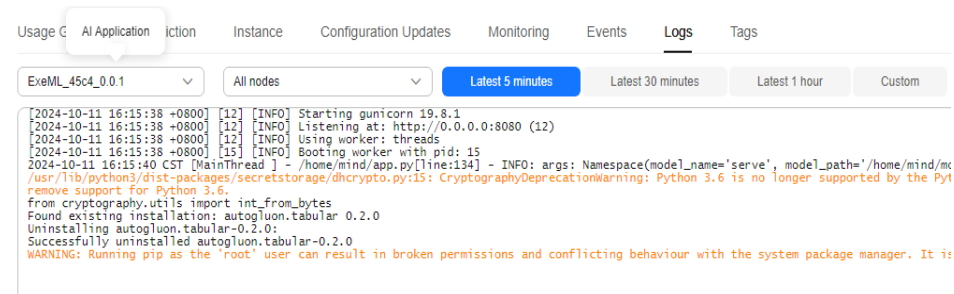
Click the model name to access its details page.

2. Deploy the AI application as a service and view service details.

On the model details page, choose **Deploy > Real-Time Services** in the upper right corner. On the **Deploy** page, select a proper instance flavor (for example, **CPU: 2 vCPUs 8 GB**), retain default settings for other parameters, and click **Next**. When the service status changes to **Running**, the service has been deployed.

Click the service name. On the displayed page, view the service details. Click the **Logs** tab to view the service logs.

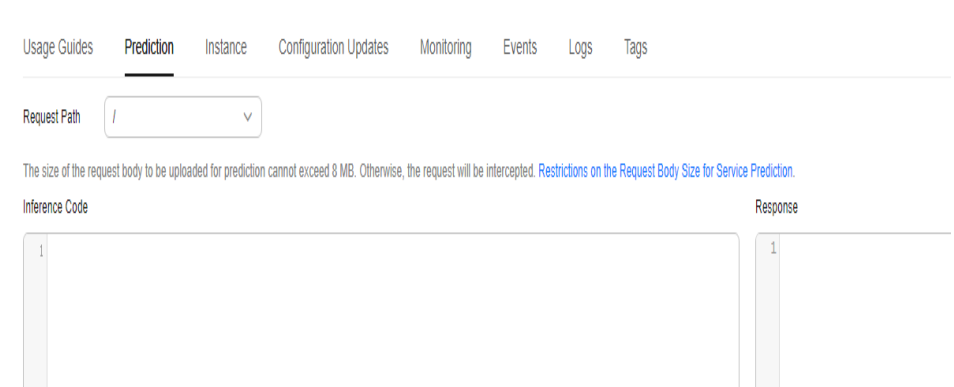
Figure 8-7 Service logs



3. Use the service for prediction.

On the service details page, click the **Prediction** tab to use the service for prediction.

Figure 8-8 Service prediction



8.3.5 Examples of Custom Scripts

To create a model in ModelArts by importing a model file from OBS, the model file package needs to comply with the ModelArts model package specifications. Additionally, the inference code and configuration file must also meet the requirements set by ModelArts.

This section provides custom script examples (including inference code examples) for common AI engines. For details about how to write model inference code, see [Specifications for Writing a Model Inference Code File](#).

Tensorflow

There are two types of TensorFlow APIs, Keras and tf. They use different code for training and saving models, but the same code for inference.

Training a Model (Keras API)

```
from keras.models import Sequential
model = Sequential()
from keras.layers import Dense
import tensorflow as tf
```



```
# Import a training dataset.
mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

print(x_train.shape)

from keras.layers import Dense
from keras.models import Sequential
import keras
from keras.layers import Dense, Activation, Flatten, Dropout

# Define a model network.
model = Sequential()
model.add(Flatten(input_shape=(28,28)))
model.add(Dense(units=5120, activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(units=10, activation='softmax'))

# Define an optimizer and loss functions.
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.summary()
# Train the model.
model.fit(x_train, y_train, epochs=2)
# Evaluate the model.
model.evaluate(x_test, y_test)
```

Saving a Model (Keras API)

```
from keras import backend as K

# K.get_session().run(tf.global_variables_initializer())

# Define the inputs and outputs of the prediction API.
# The keys of the inputs and outputs dictionaries are used as the index keys for the input and output
# tensors of the model.
# The input and output definitions of the model must match the custom inference script.
predict_signature = tf.saved_model.signature_def_utils.predict_signature_def(
    inputs={"images" : model.input},
    outputs={"scores" : model.output}
)

# Define a save path.
builder = tf.saved_model.builder.SavedModelBuilder('./mnist_keras/')

builder.add_meta_graph_and_variables(

    sess = K.get_session(),
    # The tf.saved_model.tag_constants.SERVING tag needs to be defined for inference and deployment.
    tags=[tf.saved_model.tag_constants.SERVING],
    """
signature_def_map: Only one items can exist, or the corresponding key needs to be defined as follows:
    tf.saved_model.signature_constants.DEFAULT_SERVING_SIGNATURE_DEF_KEY
    """
    signature_def_map={
        tf.saved_model.signature_constants.DEFAULT_SERVING_SIGNATURE_DEF_KEY:
            predict_signature
    }
)
builder.save()
```

Training a Model (tf API)

```
from __future__ import print_function
```

```
import gzip
import os
import urllib

import numpy
import tensorflow as tf
from six.moves import urllib

# Training data is obtained from the Yann LeCun official website http://yann.lecun.com/exdb/mnist/.
SOURCE_URL = 'http://yann.lecun.com/exdb/mnist/'
TRAIN_IMAGES = 'train-images-idx3-ubyte.gz'
TRAIN_LABELS = 'train-labels-idx1-ubyte.gz'
TEST_IMAGES = 't10k-images-idx3-ubyte.gz'
TEST_LABELS = 't10k-labels-idx1-ubyte.gz'
VALIDATION_SIZE = 5000

def maybe_download(filename, work_directory):
    """Download the data from Yann's website, unless it's already here."""
    if not os.path.exists(work_directory):
        os.mkdir(work_directory)
    filepath = os.path.join(work_directory, filename)
    if not os.path.exists(filepath):
        filepath, _ = urllib.request.urlretrieve(SOURCE_URL + filename, filepath)
        statinfo = os.stat(filepath)
        print('Successfully downloaded %s %d bytes.' % (filename, statinfo.st_size))
    return filepath

def _read32(bytestream):
    dt = numpy.dtype(numpy.uint32).newbyteorder('>')
    return numpy.frombuffer(bytestream.read(4), dtype=dt)[0]

def extract_images(filename):
    """Extract the images into a 4D uint8 numpy array [index, y, x, depth]."""
    print('Extracting %s' % filename)
    with gzip.open(filename) as bytestream:
        magic = _read32(bytestream)
        if magic != 2051:
            raise ValueError(
                'Invalid magic number %d in MNIST image file: %s' %
                (magic, filename))
        num_images = _read32(bytestream)
        rows = _read32(bytestream)
        cols = _read32(bytestream)
        buf = bytestream.read(rows * cols * num_images)
        data = numpy.frombuffer(buf, dtype=numpy.uint8)
        data = data.reshape(num_images, rows, cols, 1)
        return data

def dense_to_one_hot(labels_dense, num_classes=10):
    """Convert class labels from scalars to one-hot vectors."""
    num_labels = labels_dense.shape[0]
    index_offset = numpy.arange(num_labels) * num_classes
    labels_one_hot = numpy.zeros((num_labels, num_classes))
    labels_one_hot.flat[index_offset + labels_dense.ravel()] = 1
    return labels_one_hot

def extract_labels(filename, one_hot=False):
    """Extract the labels into a 1D uint8 numpy array [index]."""
    print('Extracting %s' % filename)
    with gzip.open(filename) as bytestream:
        magic = _read32(bytestream)
        if magic != 2049:
            raise ValueError(
                'Invalid magic number %d in MNIST label file: %s' %
```

```
(magic, filename))
num_items = _read32(bytestream)
buf = bytestream.read(num_items)
labels = numpy.frombuffer(buf, dtype=numpy.uint8)
if one_hot:
    return dense_to_one_hot(labels)
return labels

class DataSet(object):
    """Class encompassing test, validation and training MNIST data set."""

    def __init__(self, images, labels, fake_data=False, one_hot=False):
        """Construct a DataSet. one_hot arg is used only if fake_data is true."""

        if fake_data:
            self.num_examples = 10000
            self.one_hot = one_hot
        else:
            assert images.shape[0] == labels.shape[0], (
                'images.shape: %s labels.shape: %s' % (images.shape,
                                                         labels.shape))
            self.num_examples = images.shape[0]

            # Convert shape from [num examples, rows, columns, depth]
            # to [num examples, rows*columns] (assuming depth == 1)
            assert images.shape[3] == 1
            images = images.reshape(images.shape[0],
                                    images.shape[1] * images.shape[2])
            # Convert from [0, 255] -> [0.0, 1.0].
            images = images.astype(numpy.float32)
            images = numpy.multiply(images, 1.0 / 255.0)
        self._images = images
        self._labels = labels
        self._epochs_completed = 0
        self._index_in_epoch = 0

    @property
    def images(self):
        return self._images

    @property
    def labels(self):
        return self._labels

    @property
    def num_examples(self):
        return self.num_examples

    @property
    def epochs_completed(self):
        return self._epochs_completed

    def next_batch(self, batch_size, fake_data=False):
        """Return the next `batch_size` examples from this data set."""
        if fake_data:
            fake_image = [1] * 784
            if self.one_hot:
                fake_label = [1] + [0] * 9
            else:
                fake_label = 0
            return [fake_image for _ in range(batch_size)], [
                fake_label for _ in range(batch_size)
            ]
        start = self._index_in_epoch
        self._index_in_epoch += batch_size
        if self._index_in_epoch > self.num_examples:
            # Finished epoch
            self._epochs_completed += 1
```

```
# Shuffle the data
perm = numpy.arange(self._num_examples)
numpy.random.shuffle(perm)
self._images = self._images[perm]
self._labels = self._labels[perm]
# Start next epoch
start = 0
self._index_in_epoch = batch_size
assert batch_size <= self._num_examples
end = self._index_in_epoch
return self._images[start:end], self._labels[start:end]

def read_data_sets(train_dir, fake_data=False, one_hot=False):
    """Return training, validation and testing data sets."""

    class DataSets(object):
        pass

    data_sets = DataSets()

    if fake_data:
        data_sets.train = DataSet([], [], fake_data=True, one_hot=one_hot)
        data_sets.validation = DataSet([], [], fake_data=True, one_hot=one_hot)
        data_sets.test = DataSet([], [], fake_data=True, one_hot=one_hot)
        return data_sets

    local_file = maybe_download(TRAIN_IMAGES, train_dir)
    train_images = extract_images(local_file)

    local_file = maybe_download(TRAIN_LABELS, train_dir)
    train_labels = extract_labels(local_file, one_hot=one_hot)

    local_file = maybe_download(TEST_IMAGES, train_dir)
    test_images = extract_images(local_file)

    local_file = maybe_download(TEST_LABELS, train_dir)
    test_labels = extract_labels(local_file, one_hot=one_hot)

    validation_images = train_images[:VALIDATION_SIZE]
    validation_labels = train_labels[:VALIDATION_SIZE]
    train_images = train_images[VALIDATION_SIZE:]
    train_labels = train_labels[VALIDATION_SIZE:]

    data_sets.train = DataSet(train_images, train_labels)
    data_sets.validation = DataSet(validation_images, validation_labels)
    data_sets.test = DataSet(test_images, test_labels)
    return data_sets

training_iteration = 1000

modelarts_example_path = './modelarts-mnist-train-save-deploy-example'

export_path = modelarts_example_path + '/model/'
data_path = './'

print('Training model...')
mnist = read_data_sets(data_path, one_hot=True)
sess = tf.InteractiveSession()
serialized_tf_example = tf.placeholder(tf.string, name='tf_example')
feature_configs = {'x': tf.FixedLenFeature(shape=[784], dtype=tf.float32), }
tf_example = tf.parse_example(serialized_tf_example, feature_configs)
x = tf.identity(tf_example['x'], name='x') # use tf.identity() to assign name
y_ = tf.placeholder('float', shape=[None, 10])
w = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))
sess.run(tf.global_variables_initializer())
y = tf.nn.softmax(tf.matmul(x, w) + b, name='y')
cross_entropy = -tf.reduce_sum(y_ * tf.log(y))
```

```
train_step = tf.train.GradientDescentOptimizer(0.01).minimize(cross_entropy)
values, indices = tf.nn.top_k(y, 10)
table = tf.contrib.lookup.index_to_string_table_from_tensor(
    tf.constant([str(i) for i in range(10)]))
prediction_classes = table.lookup(tf.to_int64(indices))
for _ in range(training_iteration):
    batch = mnist.train.next_batch(50)
    train_step.run(feed_dict={x: batch[0], y_: batch[1]})
    correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))
    accuracy = tf.reduce_mean(tf.cast(correct_prediction, 'float'))
    print('training accuracy %g' % sess.run(
        accuracy, feed_dict={
            x: mnist.test.images,
            y_: mnist.test.labels
        }))
print('Done training!')
```

Saving a Model (tf API)

```
# Export the model.
# The model needs to be saved using the saved_model API.
print('Exporting trained model to', export_path)
builder = tf.saved_model.builder.SavedModelBuilder(export_path)

tensor_info_x = tf.saved_model.utils.build_tensor_info(x)
tensor_info_y = tf.saved_model.utils.build_tensor_info(y)

# Define the inputs and outputs of the prediction API.
# The keys of the inputs and outputs dictionaries are used as the index keys for the input and output
# tensors of the model.
# The input and output definitions of the model must match the custom inference script.
prediction_signature = (
    tf.saved_model.signature_def_utils.build_signature_def(
        inputs={'images': tensor_info_x},
        outputs={'scores': tensor_info_y},
        method_name=tf.saved_model.signature_constants.PREDICT_METHOD_NAME))

legacy_init_op = tf.group(tf.tables_initializer(), name='legacy_init_op')
builder.add_meta_graph_and_variables(
    # Set tag to serve/tf.saved_model.tag_constants.SERVING.
    sess, [tf.saved_model.tag_constants.SERVING],
    signature_def_map={
        'predict_images':
            prediction_signature,
    },
    legacy_init_op=legacy_init_op)

builder.save()

print('Done exporting!')
```

Inference Code (Keras and tf APIs)

In the model inference code file **customize_service.py**, add a child model class which inherits properties from its parent model class. For details about the parent class and import statement of each model type, see [Table 8-17](#). This example calls the parent class inference request method **_inference(self, data)**. The method does not need to be overridden in the following code.

```
from PIL import Image
import numpy as np
from model_service.tf-serving_model_service import TfServingBaseService

class MnistService(TfServingBaseService):

    # Match the model input with the user's HTTPS API input during preprocessing.
    # The model input corresponding to the preceding training part is {"images":<array>}.
    def _preprocess(self, data):
```

```
preprocessed_data = {}
images = []
# Iterate the input data.
for k, v in data.items():
    for file_name, file_content in v.items():
        image1 = Image.open(file_content)
        image1 = np.array(image1, dtype=np.float32)
        image1.resize((1,784))
        images.append(image1)
# Return the numpy array.
images = np.array(images,dtype=np.float32)
# Perform batch processing on multiple input samples and ensure that the shape is the same as that
inputted during training.
images.resize((len(data), 784))
preprocessed_data['images'] = images
return preprocessed_data

# The output corresponding to model saving in the preceding training part is {"scores":<array>}.
# Postprocess the HTTPS output.
def _postprocess(self, data):
    infer_output = {"mnist_result": []}
    # Iterate the model output.
    for output_name, results in data.items():
        for result in results:
            infer_output["mnist_result"].append(result.index(max(result)))
    return infer_output
```

Tensorflow2.1

Training and Saving a Model

```
from __future__ import absolute_import, division, print_function, unicode_literals

import tensorflow as tf

mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    # Name the output layer output, which is used to obtain the result during model inference.
    tf.keras.layers.Dense(10, activation='softmax', name="output")
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=10)

tf.keras.models.save_model(model, "./mnist")
```

Inference Code

In the model inference code file **customize_service.py**, add a child model class which inherits properties from its parent model class. For details about the parent class and import statement of each model type, see [Table 8-17](#).

```
import logging
import threading

import numpy as np
```

```
import tensorflow as tf
from PIL import Image

from model_service.tf-serving_model_service import TfServingBaseService

logger = logging.getLogger()
logger.setLevel(logging.INFO)

class MnistService(TfServingBaseService):

    def __init__(self, model_name, model_path):
        self.model_name = model_name
        self.model_path = model_path
        self.model = None
        self.predict = None

        # The label file can be loaded here and used in the post-processing function.
        # Directories for storing the label.txt file on OBS and in the model package

        # with open(os.path.join(self.model_path, 'label.txt')) as f:
        #     self.label = json.load(f)
        # Load the model in saved_model format in non-blocking mode to prevent blocking timeout.
        thread = threading.Thread(target=self.load_model)
        thread.start()

    def load_model(self):
        # Load the model in saved_model format.
        self.model = tf.saved_model.load(self.model_path)

        signature_defs = self.model.signatures.keys()

        signature = []
        # only one signature allowed
        for signature_def in signature_defs:
            signature.append(signature_def)

        if len(signature) == 1:
            model_signature = signature[0]
        else:
            logging.warning("signatures more than one, use serving_default signature from %s", signature)
            model_signature = tf.saved_model.DEFAULT_SERVING_SIGNATURE_DEF_KEY

        self.predict = self.model.signatures[model_signature]

    def _preprocess(self, data):
        images = []
        for k, v in data.items():
            for file_name, file_content in v.items():
                image1 = Image.open(file_content)
                image1 = np.array(image1, dtype=np.float32)
                image1.resize((28, 28, 1))
                images.append(image1)

        images = tf.convert_to_tensor(images, dtype=tf.dtypes.float32)
        preprocessed_data = images

        return preprocessed_data

    def _inference(self, data):
        return self.predict(data)

    def _postprocess(self, data):
        return {
            "result": int(data["output"].numpy()[0].argmax())
        }
```

Pytorch

Training a Model

```
from __future__ import print_function
import argparse
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torchvision import datasets, transforms

# Define a network structure.
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        # The second dimension of the input must be 784.
        self.hidden1 = nn.Linear(784, 5120, bias=False)
        self.output = nn.Linear(5120, 10, bias=False)

    def forward(self, x):
        x = x.view(x.size()[0], -1)
        x = F.relu((self.hidden1(x)))
        x = F.dropout(x, 0.2)
        x = self.output(x)
        return F.log_softmax(x)

def train(model, device, train_loader, optimizer, epoch):
    model.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        data, target = data.to(device), target.to(device)
        optimizer.zero_grad()
        output = model(data)
        loss = F.cross_entropy(output, target)
        loss.backward()
        optimizer.step()
        if batch_idx % 10 == 0:
            print('Train Epoch: {} [{}/{}] ({:.0f}%) \t Loss: {:.6f}'.format(
                epoch, batch_idx * len(data), len(train_loader.dataset),
                100. * batch_idx / len(train_loader), loss.item()))

def test(model, device, test_loader):
    model.eval()
    test_loss = 0
    correct = 0
    with torch.no_grad():
        for data, target in test_loader:
            data, target = data.to(device), target.to(device)
            output = model(data)
            test_loss += F.nll_loss(output, target, reduction='sum').item() # sum up batch loss
            pred = output.argmax(dim=1, keepdim=True) # get the index of the max log-probability
            correct += pred.eq(target.view_as(pred)).sum().item()

    test_loss /= len(test_loader.dataset)

    print("\nTest set: Average loss: {:.4f}, Accuracy: {}/{} ({:.0f}%) \n".format(
        test_loss, correct, len(test_loader.dataset),
        100. * correct / len(test_loader.dataset)))

device = torch.device("cpu")

batch_size=64

kwargs={}

train_loader = torch.utils.data.DataLoader(
    datasets.MNIST('.', train=True, download=True,
        transform=transforms.Compose([
            transforms.ToTensor()
```



```
    ])),
    batch_size=batch_size, shuffle=True, **kwargs)
test_loader = torch.utils.data.DataLoader(
    datasets.MNIST('.', train=False, transform=transforms.Compose([
        transforms.ToTensor()
    ])),
    batch_size=1000, shuffle=True, **kwargs)

model = Net().to(device)
optimizer = optim.SGD(model.parameters(), lr=0.01, momentum=0.5)
optimizer = optim.Adam(model.parameters())

for epoch in range(1, 2 + 1):
    train(model, device, train_loader, optimizer, epoch)
    test(model, device, test_loader)
```

Saving a Model

```
# The model must be saved using state_dict and can be deployed remotely.
torch.save(model.state_dict(), "pytorch_mnist/mnist_mlp.pt")
```

Inference Code

In the model inference code file **customize_service.py**, add a child model class which inherits properties from its parent model class. For details about the parent class and import statement of each model type, see [Table 8-17](#).

```
from PIL import Image
import log
from model_service.pytorch_model_service import PTServingBaseService
import torch.nn.functional as F

import torch.nn as nn
import torch
import json

import numpy as np

logger = log.getLogger(__name__)

import torchvision.transforms as transforms

# Define model preprocessing.
infer_transformation = transforms.Compose([
    transforms.Resize((28,28)),
    # Convert data to tensor.
    transforms.ToTensor()
])

import os

class PTVisionService(PTServingBaseService):

    def __init__(self, model_name, model_path):
        # Call the constructor of the parent class.
        super(PTVisionService, self).__init__(model_name, model_path)
        # Call the custom function to load the model.
        self.model = Mnist(model_path)
        # Load labels.
        self.label = [0,1,2,3,4,5,6,7,8,9]
        # Labels can also be loaded by label file.
        # Reads the label.json file in the model directory.
        dir_path = os.path.dirname(os.path.realpath(self.model_path))
        with open(os.path.join(dir_path, 'label.json')) as f:
            self.label = json.load(f)
```

```
def _preprocess(self, data):

    preprocessed_data = {}
    for k, v in data.items():
        input_batch = []
        for file_name, file_content in v.items():
            with Image.open(file_content) as image1:
                # Gray processing
                image1 = image1.convert("L")
                if torch.cuda.is_available():
                    input_batch.append(infer_transformation(image1).cuda())
                else:
                    input_batch.append(infer_transformation(image1))
            input_batch_var = torch.autograd.Variable(torch.stack(input_batch, dim=0), volatile=True)
            print(input_batch_var.shape)
            preprocessed_data[k] = input_batch_var

    return preprocessed_data

def _postprocess(self, data):
    results = []
    for k, v in data.items():
        result = torch.argmax(v[0])
        result = {k: self.label[result]}
        results.append(result)
    return results

def _inference(self, data):

    result = {}
    for k, v in data.items():
        result[k] = self.model(v)

    return result

class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.hidden1 = nn.Linear(784, 5120, bias=False)
        self.output = nn.Linear(5120, 10, bias=False)

    def forward(self, x):
        x = x.view(x.size()[0], -1)
        x = F.relu((self.hidden1(x)))
        x = F.dropout(x, 0.2)
        x = self.output(x)
        return F.log_softmax(x)

def Mnist(model_path, **kwargs):
    # Generate a network.
    model = Net()
    # Load the model.
    if torch.cuda.is_available():
        device = torch.device('cuda')
        model.load_state_dict(torch.load(model_path, map_location="cuda:0"))
    else:
        device = torch.device('cpu')
        model.load_state_dict(torch.load(model_path, map_location=device))
    # CPU or GPU mapping
    model.to(device)
    # Set the model to evaluation mode.
    model.eval()

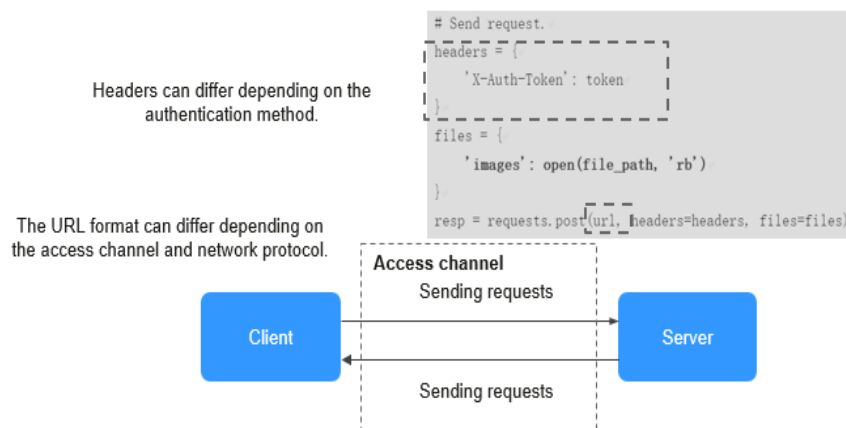
    return model
```

8.4 Deploying a Model as Real-Time Inference Jobs

8.4.1 Deploying and Using Real-Time Inference

After creating a model, you can deploy it as a real-time service. If a real-time service is in the Running status, it has been deployed. This service provides a standard, callable RESTful API. When accessing a real-time service, you can choose the authentication method, access channel, and transmission protocol that best suit your needs. These three elements make up your access requests and can be mixed and matched without any interference. For example, you can use different authentication methods for different access channels and transmission protocols.

Figure 8-9 Authentication method, access channel, and transmission protocol



ModelArts supports the following authentication methods for accessing real-time services (HTTPS requests are used as examples):

- **Token-based authentication:** The validity period of a token is 24 hours. When using a token for authentication, cache it to prevent frequent calls.
- **AK/SK-based authentication:** AK/SK is used to sign requests and the signature is then added to the request for authentication. AK/SK-based authentication supports API requests with a body not larger than 12 MB. For API requests with a larger body, token-based authentication is recommended.
- **App authentication:** Add a parameter to the request header to complete the authentication. The authentication is simple and permanently valid.

ModelArts allows you to call APIs to access real-time services in the following ways (HTTPS requests are used as examples):

- **Accessing a Real-Time Service Through a Public Network:** By default, ModelArts inference uses the public network to access real-time services. A standard, callable RESTful API is provided after deployment of a real-time service.
- **Accessing a Real-Time Service Through a VPC High-Speed Channel:** When using VPC peering for high-speed access, your service requests are sent

directly to instances via VPC peering, bypassing the inference platform. This results in faster service access.

Real-time service APIs are accessed using HTTPS by default. Additionally, the following transmission protocols are also supported:

- **Accessing a Real-Time Service Using WebSocket:** WebSocket simplifies data exchange between the client and server and allows the server to proactively push data to the client. In the WebSocket API, if the initial handshake between the client and the server is successful, a persistent connection will be established between them and data can be transferred bidirectionally.
- **Accessing a Real-Time Service Using Server-Sent Events:** Server-Sent Events (SSE) primarily facilitates unidirectional real-time communication from the server to the client, such as streaming ChatGPT responses. In contrast to WebSockets, which provide bidirectional real-time communication, SSE is designed to be more lightweight and simpler to implement.

8.4.2 Deploying a Model as a Real-Time Service

Real-time inference involves taking user inputs or queries via the internet and instantly providing processed outcomes or choices using AI or machine learning models hosted on remote servers or cloud platforms. Real-time inference uses cloud-based models to deliver fast and reliable AI services. It offers powerful analysis, predictions, and understanding without needing local model deployments. This solution works best for tasks requiring quick responses and interactions.

ModelArts allows you to deploy a model as a web service that provides a real-time test UI and monitoring capabilities. The deployed real-time service offers an accessible API for predictions and calls. Real-time inference is used in situations that need fast responses, like online intelligent customer service and autonomous driving decisions.

This section describes how to deploy your model as a real-time service on ModelArts and use it for predictions.

Billing

Deploying a service in ModelArts uses compute and storage resources, which are billed. Compute resources are billed for running the inference service. Storage resources are billed for storing data in OBS. For details, see [Table 8-19](#).

Table 8-19 Billing items

Billing Item		Description	Billing Mode	Billing Formula
Com pute reso urces	Publ ic reso urce pool	Usage of compute resources. For details, see ModelArts Pricing Details .	Pay-per-use	Specification unit price x Number of compute nodes x Usage duration

Billing Item		Description	Billing Mode	Billing Formula
	Dedicated resource pool	Fees for dedicated resource pools are paid upfront upon purchase. There are no additional charges for service deployment. For details about dedicated resource pool fees, see Dedicated Resource Pool Billing Items .	N/A	N/A
Event notification (billed only when enabled)		This function uses Simple Message Notification (SMN) to send a message to you when the event you selected occurs. To use this function, enable event notification when creating a training job. For pricing details, see SMN Pricing Details .	Pay by actual usage	• SMS: SMS notifications • Email: Email notifications + Downstream Internet traffic • HTTP or HTTPS: HTTP or HTTPS notifications + Downstream Internet traffic
Run logs (billed only when enabled)		Log Tank Service (LTS) collects, analyzes, and stores logs. If Runtime Log Output is enabled during service deployment, you will be billed if the log data exceeds the LTS free quota. For details, see Log Tank Service Pricing Details .	Pay by actual log size	After the free quota is exceeded, you are billed based on the actual log volume and retention duration.

Constraints

A user can create up to 20 real-time services.

Prerequisites

- A ModelArts model in the **Normal** state is available. For details about how to create models, see [Creating a Model](#).
- The account is not in arrears to ensure available resources for service running.
- To mount SFS Turbo to a real-time service, first create an SFS Turbo file system and associate it with the service. Follow these steps:
 - a. Create an SFS Turbo file system. For details, see [Create a File System](#).

- b. On the **Standard Cluster** page, click the resource pool where you want to deploy the service. Copy the value of the **Network** field and exit the details page
- c. Exit the details page. Click the **Network** tab and search for the target network using the copied information. Click **Interconnect VPC**, select the VPC and subnet where your SFS Turbo is located, and click **OK**.

Alternatively, choose **More > Add sfsturbo** and select the SFS Turbo file system you want to mount. In this step, the ECS specifications of the SFS Turbo file system must support multiple NICs. Otherwise, attaching NICs fails.

Deploying a Real-Time Service (Synchronous Request)

1. Log in to the [ModelArts console](#). In the navigation pane, choose **Model Deployment > Real-Time Services**.
2. In the real-time service list, click **Deploy** in the upper left corner.
3. Configure parameters.
 - a. Configure basic parameters. For details, see [Table 8-20](#).

Table 8-20 Basic parameters

Parameter	Description
Name	Name of a real-time service.
Auto Stop	Time for your service to automatically stop running. This helps you avoid unnecessary billing. If you disable this feature, your real-time service will continue running and you will be billed accordingly. By default, this feature is enabled and set to stop the service 1 hour after it starts. The options are 1 hour , 2 hours , 4 hours , 6 hours , and Custom . If you select Custom , you can enter any integer from 1 to 24.
Description	Brief description for a real-time service.

- b. Enter key information including the resource pool and model configurations. For details, see [Table 8-21](#).

Table 8-21 Parameters

Parameter	Sub-Parameter	Description
Resource Pool	Public Resource Pool	Public resource pool for deploying the real-time service. Public resource pools provide large-scale public computing clusters, which are allocated based on job parameter settings. Resources are isolated by job. CPU/GPU resource pools are available for you to select. The pricing for resource pools varies depending on their flavors. For details, see Product Pricing Details . Public resource pools only support the pay-per-use billing mode.
	Dedicated Resource Pool	Dedicated resource pool for deploying the real-time service. The resources provided in a dedicated resource pool are exclusive and more controllable. Select a dedicated resource pool flavor. The physical pools with logical subpools created are not supported temporarily.
Model and Configuration	Model Source	Model source for deploying the real-time service. Choose My Model or My Subscriptions as needed. • My Model: Models either trained on ModelArts or developed elsewhere and then uploaded to ModelArts. • My Subscriptions: Models subscribed from AI Gallery.
	Model and Version	Model and version that are in the Normal state.
	Traffic Ratio (%)	Traffic percentage of the current model version. Service call requests are allocated to the current version based on this proportion. If you deploy only one version of a model, set this parameter to 100 . If you select multiple versions for gray release, ensure that the sum of the traffic ratios of these versions is 100% .

Parameter	Sub-Parameter	Description
	Instance Flavor	Instance flavor of the real-time service to ensure smooth operation. Select available flavors based on the list displayed on the console. The flavors in gray cannot be used in the current environment. If no public resource pool flavors are available, use a dedicated resource pool. When deploying the service with the selected flavor, there will be necessary system consumptions. This means that the actual resources required will be greater than the flavor.
	Instances	Number of instances for the current model version. If you set the number of instances to 1, the standalone computing mode is used. If you set the number of instances to a value greater than 1, the distributed computing mode is used. Select a computing mode based on your actual needs.
	Environment Variable	Environment variables you need to inject to the pod. To ensure data security, do not enter sensitive information, such as plaintext passwords, in environment variables.
	Timeout	Timeout of a single model, including both the deployment and startup time. The default value is 20 minutes. The value must range from 3 to 120.
	Add Model and Configuration	If the selected model has multiple versions, you can add multiple versions and configure traffic ratios for each version. You can use gray release to smoothly upgrade the model version. Free compute specifications do not support gray release of multiple versions.

Parameter	Sub-Parameter	Description
	Mount Storage	This parameter is displayed when the resource pool is a dedicated resource pool. This feature will mount a storage volume to compute nodes (instances) as a local directory when the service is running. This is a good option to consider when dealing with large input data or models. SFS Turbo Preparations for mounting SFS Turbo: Storage mounting is allowed only for services deployed in a dedicated resource pool which has interconnected with a VPC or associated with SFS Turbo. • To interconnect a VPC is to interconnect the VPC where SFS Turbo belongs to a dedicated resource pool network. For details, see Interconnect with a VPC. • You can associate HPC SFS Turbo file systems with dedicated resource pool networks. Parameters: • File System Name: Select the target SFS Turbo file system. A cross-region SFS Turbo file system cannot be selected. • Mount Path: Enter the mount path of the container, for example, <code>/sfs-turbo-mount/</code>. Select a new directory. If you select an existing directory, any existing files within it will be replaced. Notes: • A file system can be mounted only once and to only one path. Each mount path must be unique. A maximum of 8 disks can be mounted to a training job. • If you need to mount multiple file systems, do not use same or similar paths, for example, <code>/obs-mount/</code> and <code>/obs-mount/tmp/</code>. • Once you have chosen SFS Turbo, avoid deleting the interconnected VPC or disassociating SFS Turbo. Otherwise, mounting will not be possible. When you mount the backend OBS storage on the SFS Turbo page, make sure to set the client's umask permission to 777 for normal use.

Parameter	Sub-Parameter	Description
Priority	N/A	This function is supported only for dedicated resource pools, including logical resource pools, new physical resource pools, and logical subpools. Existing physical resource pools do not support this function. You can set this parameter to preferentially schedule high-priority services. Priority values range from 1 (lowest) to 3 (highest). When training and inference jobs share the same pool, and Preemption is enabled on the training job creation page (For details, see Creating a Production Training Job (New Version)), an inference task with a higher priority can preempt a training job with a lower priority.
Traffic Limit	N/A	Maximum number of times a service can be accessed within a second. You can configure this parameter as needed.
WebSocket	N/A	Specifies whether to deploy a real-time service as a WebSocket service. Change the communication protocol of the service from HTTP/HTTPS to WebSocket. WebSocket enables bidirectional, instant communication between clients and servers, making it ideal for applications like real-time predictions and chatbot interactions. Once the protocol switches to WebSocket, the service's API URL updates to a WebSocket address. Clients can then link to the service and share data using a WebSocket client. Constraints: This feature is supported only if the model is WebSocket-compliant and comes from a container image. After this feature is enabled, Traffic Limit and Data Collection cannot be set. This parameter cannot be modified after the service is deployed. For details about WebSocket real-time services, see Full-Process Development of WebSocket Real-Time Services.

Parameter	Sub-Parameter	Description
Application Authentication	Application	Specifies whether to control access to the real-time service through app authentication. App authentication verifies a client's identity using their AppCode and AppSecret. It allows only authorized apps to access service APIs. App authentication allows better access control and boosts service security. This feature is disabled by default. To enable this feature, see Accessing a Real-Time Service Through App Authentication for details and configure parameters as required.

- c. (Optional) Configure advanced settings.

Table 8-22 Advanced settings

Parameter	Description
Auto Restart	Specifies whether to automatically restart a service instance when a fault occurs. After this function is enabled, the system automatically redeploys the real-time service when detecting that the real-time service is abnormal. For details, see Configuring Auto Restart upon a Real-Time Service Fault. Auto restart boosts service reliability, minimizes downtime, and handles hardware failures efficiently. Use this function for tasks needing reliable and stable performance.
Tags	ModelArts can work with Tag Management Service (TMS). When creating resource-consuming tasks in ModelArts, for example, training jobs, configure tags for these tasks so that ModelArts can use tags to manage resources by group. You can select a predefined TMS tag from the tag drop-down list or customize a tag. Predefined tags are available to all service resources that support tags. Custom tags are available only to the service resources of the user who has created the tags. For details about how to use tags, see Using TMS Tags to Manage Resources by Group

4. After confirming the entered information, deploy the service as prompted. Deploying a service generally requires a period of time, which may be several

minutes or tens of minutes depending on the amount of your data and resources.

You can go to the real-time service list to check if the deployment is complete. Once the service status changes from **Deploying** to **Running**, the service is deployed.

Once a real-time service is deployed, it will start immediately.

Testing Real-Time Service Prediction

After a model is deployed as a real-time service, you can debug code or add files for testing in the **Prediction** tab. Due to the limitation of API Gateway, the duration of a single prediction cannot exceed 40s.

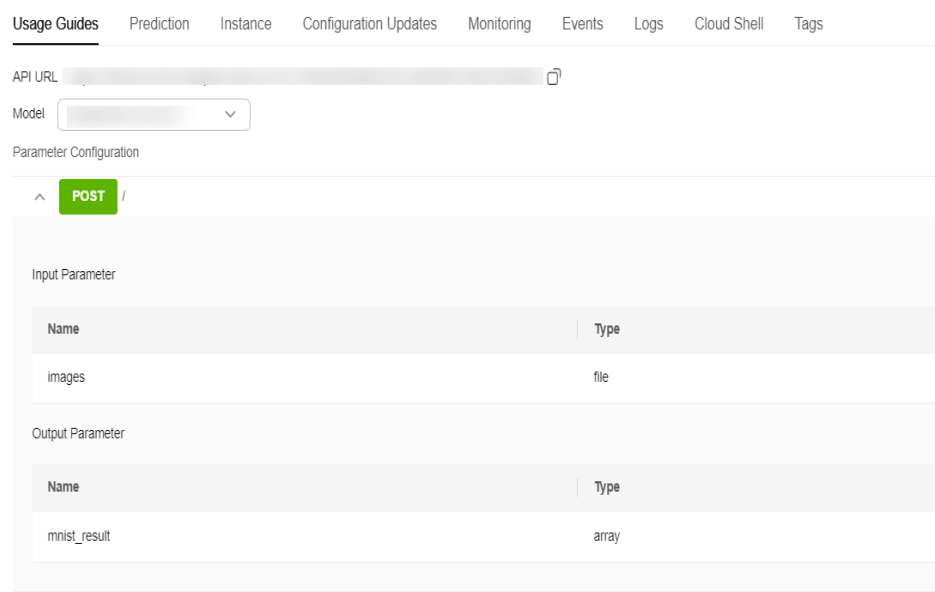
This feature is used for commissioning. Use API calling for actual production. You can select [Accessing a Real-Time Service Through Token-based Authentication](#), [Accessing a Real-Time Service Through AK/SK-based Authentication](#), or [Accessing a Real-Time Service Through App Authentication](#) based on the authentication method.

You can test the service in two ways, depending on the input request defined by the model – either by using a JSON text or a file.

- **JSON text prediction:** If the input of the deployed model is JSON text, you can enter JSON code in the **Prediction** tab for testing.
- **File Prediction:** If your model uses files as input, you can add images, audios, or videos into the **Prediction** tab to test the service.
 - The size of an input image must be less than 8 MB.
 - The maximum size of a request body for JSON text prediction is 8 MB.
 - Due to the limitation of API Gateway, the duration of a single prediction cannot exceed 40s.
 - The following image types are supported: png, psd, jpg, jpeg, bmp, gif, webp, psd, svg, and tiff.
 - If you use Ascend flavors for service deployment, you cannot predict transparent .png images because Ascend only supports RGB-3 images.

After a service is deployed, obtain the input parameters of the service in the **Usage Guides** page of the service details page.

Figure 8-10 Usage Guides



The input parameters displayed in the **Usage Guides** tab depend on the model source that you select.

- If your meta model comes from a built-in algorithm, the input and output parameters are defined by ModelArts. For details, see the **Usage Guides** tab. In the **Prediction** tab, enter the corresponding JSON text or file for service testing.
- If you use a custom meta model and your own inference code and configuration file (see [Specifications for Writing the Model Configuration File](#)), the **Usage Guides** tab will only display your configuration file. The following figure shows the mapping between the input parameters in the **Usage Guides** tab and the configuration file.

Figure 8-11 Mapping between the configuration file and Usage Guides

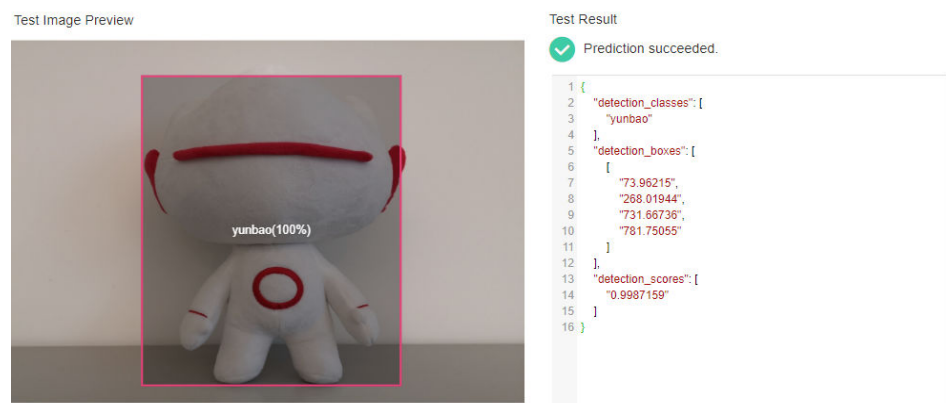


The prediction methods for different input requests are as follows:

- **JSON Text Prediction**
 - a. Log in to the [ModelArts console](#) and choose **Model Deployment > Real-Time Services**.

- b. Click the name of the target service to access its details page. Enter the inference code in the **Prediction** tab, and click **Predict** to perform prediction.
- **File Prediction**
 - a. Log in to the [ModelArts console](#) and choose **Model Deployment > Real-Time Services**.
 - b. Click the name of the target service to access its details page. In the **Prediction** tab, click **Upload** and select a test file. After the file is uploaded, click **Predict** to perform a prediction test. In [Figure 8-12](#), the label, position coordinates, and confidence score are displayed.

Figure 8-12 Image prediction



Using Cloud Shell to Debug a Real-Time Service Instance Container

You can use Cloud Shell provided by the ModelArts console to log in to the instance container of a running real-time service.

Constraints:

- Cloud Shell can only access a container when the associated real-time service is deployed within a dedicated resource pool
- Cloud Shell can only access a container when the associated real-time service is running.

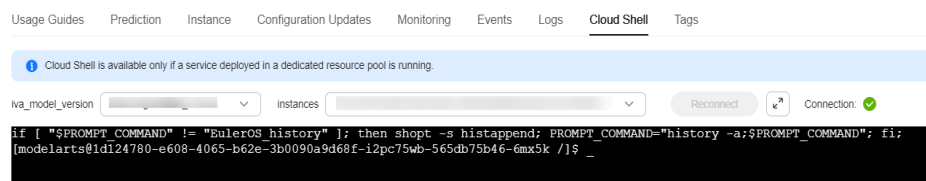
Step 1 Log in to the [ModelArts console](#). In the navigation pane, choose **Model Deployment > Real-Time Services**.

Step 2 On the real-time service list page, click the name or ID of the target service.

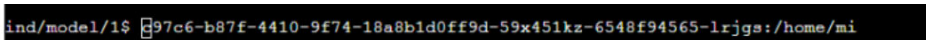
Step 3 Click the **Cloud Shell** tab and select the target model version and compute node.

When the connection status changes to , you have logged in to the instance container.

If the server disconnects due to an error or remains idle for 10 minutes, you can select **Reconnect** to regain access to the pod.

Figure 8-13 Cloud Shell

If you encounter a path display issue when logging in to Cloud Shell, press **Enter** to resolve the problem.

Figure 8-14 Path display issue

Step 4 After logging in to the container, execute the necessary debugging commands in its terminal. Example:

View logs:

```
tail -f /var/log/app.log
```

Check the service status:

```
systemctl status app
```

Run a custom script:

```
./debug_script.sh
```

Step 5 After the debugging, exit the container:

```
exit
```

After returning to the Cloud Shell terminal, you can view the debugging result or log file.

----End

FAQs

- Service deployment failed
 - [Alarm Status of a Deployed Real-Time Service](#)
 - [Error Occurred When a Custom Image Model Is Deployed as a Real-Time Service](#)
 - [Service Is Consistently Being Deployed](#)
- Service prediction failed
 - [Service Prediction Failed](#)
 - [Error "APIG.XXXX" Occurred in a Prediction Failure](#)

8.4.3 Authentication Methods for Accessing Real-time Services

8.4.3.1 Accessing a Real-Time Service Through Token-based Authentication

A token specifies certain permissions in a computer system. During token-based authentication, the token is added to requests to get permissions for calling the API.

If a real-time service is in the **Running** state, it has been deployed successfully. This service provides a standard RESTful API for users to call. You can use token-based authentication to verify your identity when calling a real-time service API.

ModelArts uses tokens for accessing real-time services. Tokens typically last 24 hours. Once expired, you must get a new one. This method suits fast development and testing due to its simplicity, though tokens expire quickly.

Test the real-time service's RESTful API before deploying it to production. Send an inference request using token-based authentication with these steps:

- **Method 1: Use GUI-based Software for Inference (Postman).** (Postman is recommended for Windows.)
- **Method 2: Run the cURL Command to Send an Inference Request.** (curl commands are recommended for Linux.)
- **Method 3: Use Python to Send an Inference Request.**
- **Method 4: Use Java to Send an Inference Request.**

Constraints

When you call an API to access a real-time service, the size of the prediction request body and the prediction time are subject to the following limitations:

- The size of a request body cannot exceed 12 MB. Otherwise, the request will fail.
- Due to the limitation of API Gateway, the prediction duration of each request does not exceed 40 seconds.

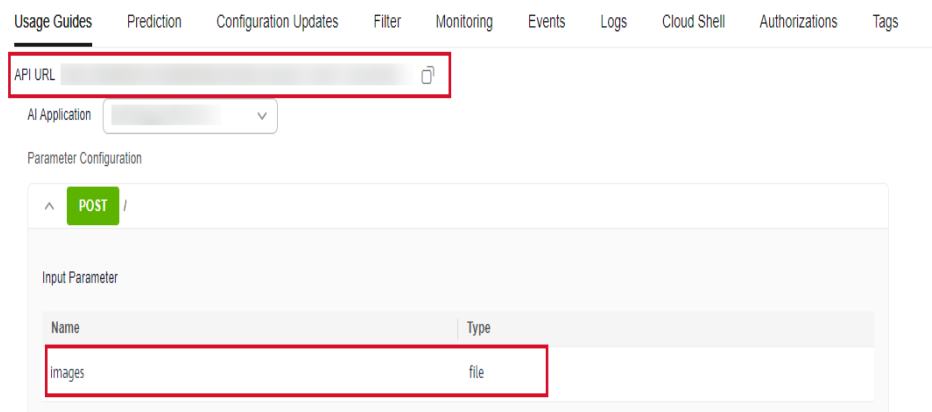
Prerequisites

You have obtained a user token, local path to the inference file, URL of the real-time service, and input parameters of the real-time service.

- For details about how to obtain a user token, see **Token-based Authentication**. The real-time service APIs generated by ModelArts do not support tokens whose scope is domain. Therefore, you need to obtain the token whose scope is project.
- The local path to the inference file can be an absolute path (for example, **D:/test.png** for Windows and **/opt/data/test.png** for Linux) or a relative path (for example, **./test.png**).
- You can obtain the service URL and input parameters of a real-time service on the Usage Guides tab page of its service details page.

The API URL is the service URL of the real-time service. If a path is defined for **apis** in the model configuration file, the URL must be followed by the user-defined path, for example, **{URL of the real-time service}/predictions/poetry**.

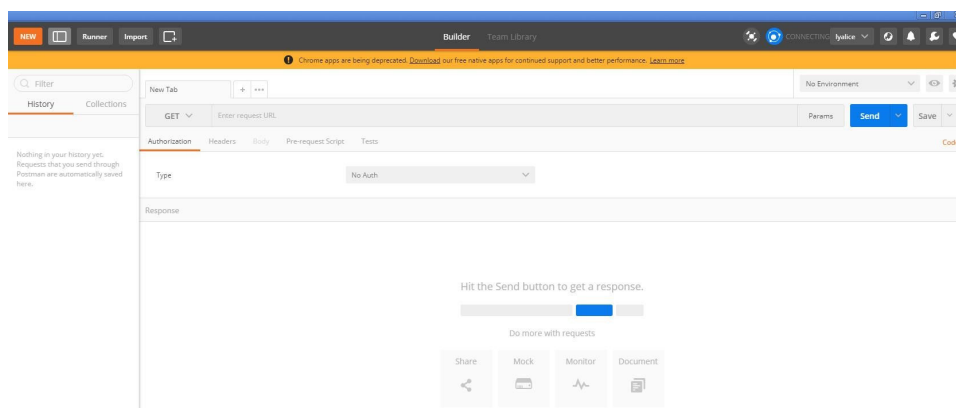
Figure 8-15 Obtaining the API URL and file prediction input parameters of a real-time service



Method 1: Use GUI-based Software for Inference (Postman)

1. Download Postman and install it, or install the Postman Chrome extension. Alternatively, use other software that can send POST requests. Postman 7.24.0 is recommended.
2. Open Postman. [Figure 8-16](#) shows the Postman interface.

Figure 8-16 Postman interface



3. Set parameters on Postman. The following uses image classification as an example.
 - Select a POST task and copy the API URL to the POST text box. On the **Headers** tab page, set **Key** to **X-Auth-Token** and **Value** to the user token.

NOTE

You can also use the AK and SK to encrypt API calling requests. For details, see [Overview of Session Authentication](#).

Figure 8-17 Parameter settings

KEY	VALUE
X-Auth-Token	MIITQAYJKoZIhvcNAQcCollTMTCCeY0CAQExDTALBgghkgBZQMEAgEwghG0BgkqhkiG9w...

- In the **Body** tab, file input and text input are available.

- **File input**

Select **form-data**. Set **KEY** to the input parameter of the model, which must be the same as the input parameter of the real-time service. In this example, the **KEY** is **images**. Set **VALUE** to an image to be inferred (only one image can be inferred). See [Figure 8-18](#).

Figure 8-18 Setting parameters on the **Body** tab page

KEY	VALUE
images	Choose Files image_0001.jpg

- **Text input**

Select **raw** and then **JSON(application/json)**. Enter the request body in the text box below. An example request body is as follows:

```
{
  "meta": {
    "uuid": "10eb0091-887f-4839-9929-cbc884f1e20e"
  },
  "data": {
    "req_data": [
      {
        "sepal_length": 3,
        "sepal_width": 1,
        "petal_length": 2.2,
        "petal_width": 4
      }
    ]
  }
}
```

meta can carry a universally unique identifier (UUID). When you call an API, the system provides a UUID. When the inference result is returned, the UUID is returned to trace the request. If you do not need this function, leave **meta** blank. **data** contains a **req_data** array for one or multiple pieces of input data. The parameters of each piece of data are determined by the model, such as **sepal_length** and **sepal_width** in this example.

4. After setting the parameters, click **send** to send the request. The result will be displayed in **Response**.
 - Inference result using file input: [Figure 8-19](#) shows an example. The field values in the return result vary with the model.
 - Inference result using text input: [Figure 8-20](#) shows an example. The request body contains **meta** and **data**. If the request contains **uuid**, **uuid**

will be returned in the response. Otherwise, **uuid** is left blank. **data** contains a **resp_data** array for the inference results of one or multiple pieces of input data. The parameters of each result are determined by the model, for example, **sepal_length** and **predictresult** in this example.

Figure 8-19 File inference result

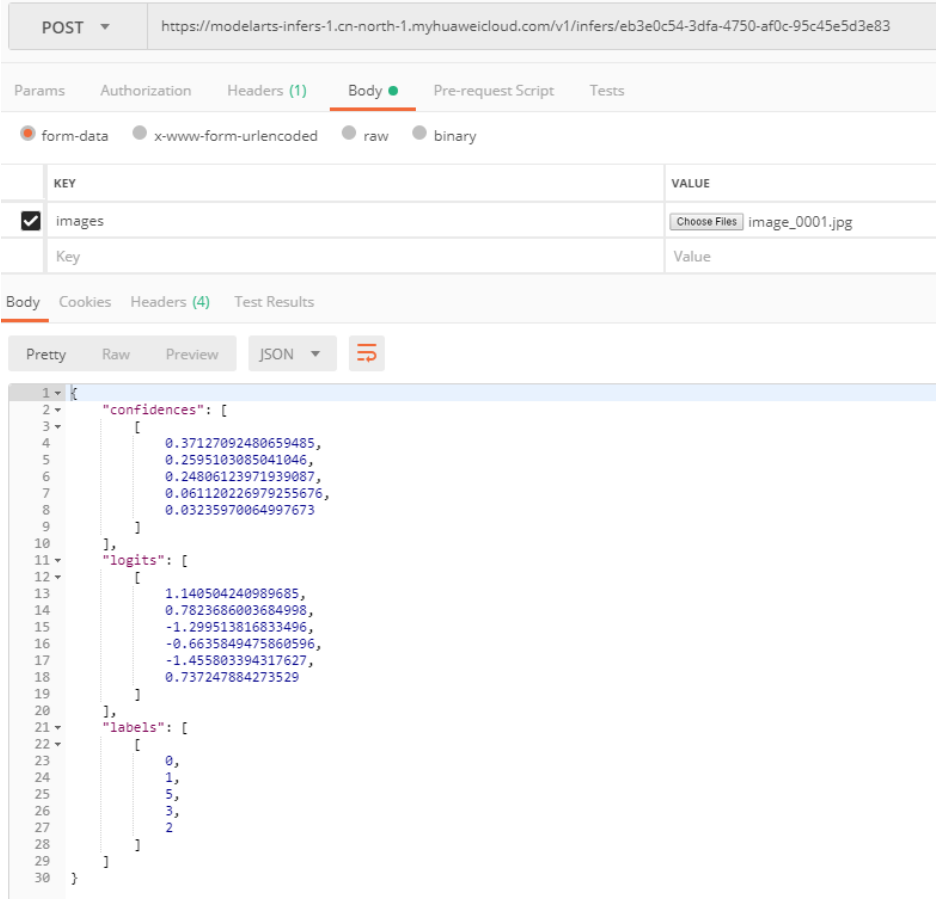
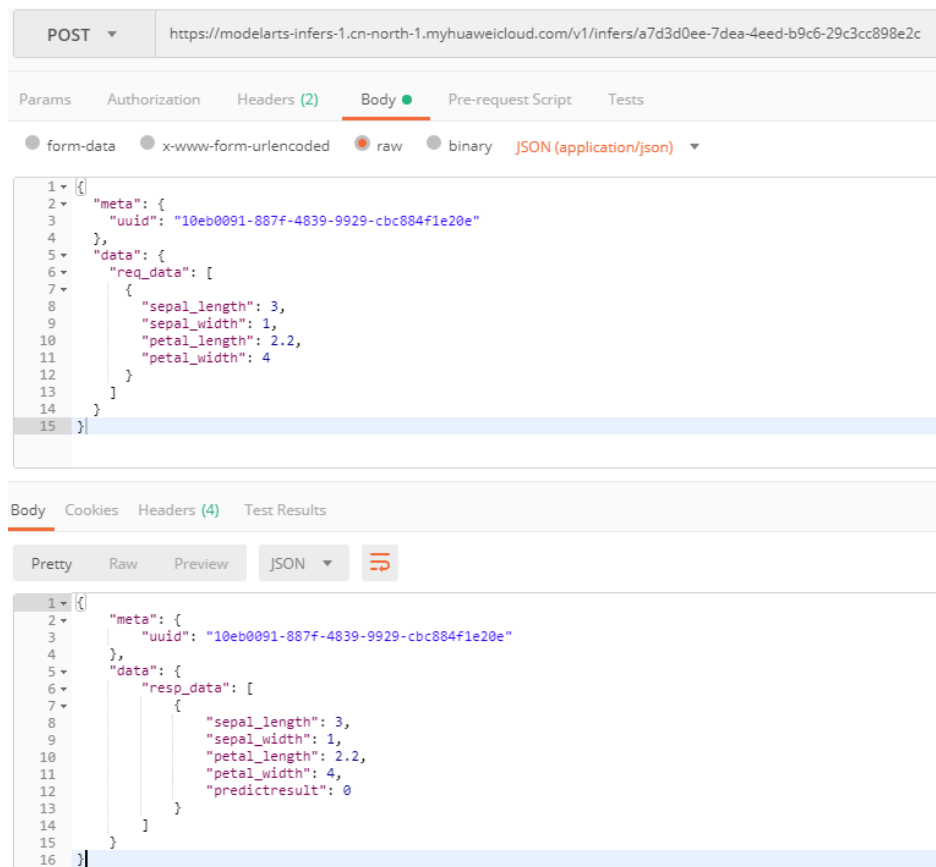


Figure 8-20 Text inference result

Method 2: Run the cURL Command to Send an Inference Request

The command for sending inference requests can be input as a file or text.

- File input

```
curl -kv -F 'images=@Image path' -H 'X-Auth-Token:Token value' -X POST Real-time service URL
```

- **-k** indicates that SSL websites can be accessed without using a security certificate.
- **-F** indicates file input. In this example, the parameter name is **images**, which can be changed as required. The image storage path follows **@**.
- **-H** indicates the header of a POST command. **X-Auth-Token** is the header key, which is fixed. *Token value* indicates the user token.
- **POST** is followed by the API URL of the real-time service.

The following is an example of the cURL command for inference with file input:

```
curl -kv -F 'images=@/home/data/test.png' -H 'X-Auth-Token:MIISkAY***80T9wHQ==' -X POST https://modelarts-infers-1.xxx/v1/infers/eb3e0c54-3dfa-4750-af0c-95c45e5d3e83
```

- Text input

```
curl -kv -d '{"data":{"req_data": [{"sepal_length":3,"sepal_width":1,"petal_length":2.2,"petal_width":4}]}}' -H 'X-Auth-Token:MIISkAY***80T9wHQ==' -H 'Content-type: application/json' -X POST https://modelarts-infers-1.xxx/v1/infers/eb3e0c54-3dfa-4750-af0c-95c45e5d3e83
```

-**d** indicates the text input of the request body.

Method 3: Use Python to Send an Inference Request

1. Download the Python SDK and configure it in the development tool. For details, see [Integrating the Python SDK for API request signing](#).
2. Create a request body for inference.

- **File input**

```
# coding=utf-8

import requests

if __name__ == '__main__':
    # Config url, token and file path.
    url = "URL of the real-time service"
    token = "User token"
    file_path = "Local path to the inference file"

    # Send request.
    headers = {
        'X-Auth-Token': token
    }
    files = {
        'images': open(file_path, 'rb')
    }
    resp = requests.post(url, headers=headers, files=files)

    # Print result.
    print(resp.status_code)
    print(resp.text)
```

The **files** name is determined by the input parameter of the real-time service. The parameter name must be the same as that of the input parameter of the file type. The input parameter **images** obtained in [Prerequisites](#) is an example.

- **Text input (JSON)**

The following is an example of the request body for reading the local inference file and performing Base64 encoding:

```
# coding=utf-8

import base64
import requests

if __name__ == '__main__':
    # Config url, token and file path
    url = "URL of the real-time service"
    token = "User token"
    file_path = "Local path to the inference file"
    with open(file_path, "rb") as file:
        base64_data = base64.b64encode(file.read()).decode("utf-8")

    # Set body, then send request
    headers = {
        'Content-Type': 'application/json',
        'X-Auth-Token': token
    }
    body = {
        'image': base64_data
    }
    resp = requests.post(url, headers=headers, json=body)

    # Print result
    print(resp.status_code)
    print(resp.text)
```

The **body** name is determined by the input parameter of the real-time service. The parameter name must be the same as that of the input

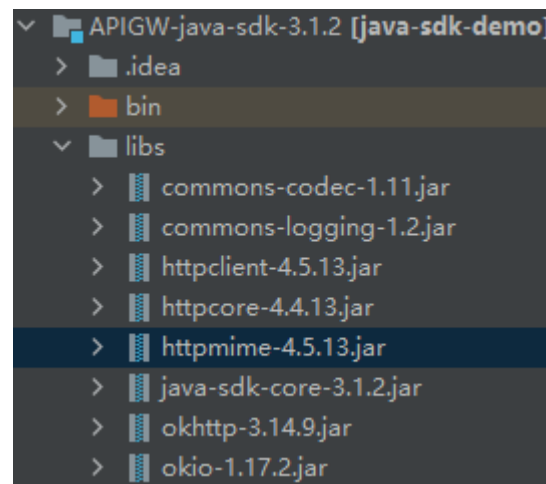
parameter of the string type. The input parameter **images** obtained in [Prerequisites](#) is an example. The value of **base64_data** in **body** is of the string type.

Method 4: Use Java to Send an Inference Request

1. Download the Java SDK and configure it in the development tool. For details, see [Integrating the Java SDK for API request signing](#).
2. (Optional) If the input of the inference request is in the file format, the Java project depends on the httpmime module.
 - a. Add **httpmime-x.x.x.jar** to the **libs** folder. [Figure 8-21](#) shows a complete Java dependency library.

You are advised to use httpmime-x.x.x.jar 4.5 or later. Download httpmime-x.x.x.jar from <https://mvnrepository.com/artifact/org.apache.httpcomponents/httpmime>.

Figure 8-21 Java dependency library



- b. After **httpmime-x.x.x.jar** is added, add httpmime information to the **.classpath** file of the Java project as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<classpath>
<classpathentry kind="con" path="org.eclipse.jdt.launching.JRE_CONTAINER"/>
<classpathentry kind="src" path="src"/>
<classpathentry kind="lib" path="libs/commons-codec-1.11.jar"/>
<classpathentry kind="lib" path="libs/commons-logging-1.2.jar"/>
<classpathentry kind="lib" path="libs/httpclient-4.5.13.jar"/>
<classpathentry kind="lib" path="libs/httpcore-4.4.13.jar"/>
<classpathentry kind="lib" path="libs/httpmime-x.x.x.jar"/>
<classpathentry kind="lib" path="libs/java-sdk-core-3.1.2.jar"/>
<classpathentry kind="lib" path="libs/okhttp-3.14.9.jar"/>
<classpathentry kind="lib" path="libs/okio-1.17.2.jar"/>
<classpathentry kind="output" path="bin"/>
</classpath>
```

3. Create a Java request body for inference.
 - **File input**

A sample Java request body is as follows:

```
// Package name of the demo.
package com.apigw.sdk.demo;

import org.apache.http.Consts;
```

```
import org.apache.http.HttpEntity;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.ContentType;
import org.apache.http.entity.mime.MultipartEntityBuilder;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;

import java.io.File;

public class MyTokenFile {

    public static void main(String[] args) {
        // Config url, token and filePath
        String url = "URL of the real-time service";
        String token = "User token";
        String filePath = "Local path to the inference file";

        try {
            // Create post
            HttpPost httpPost = new HttpPost(url);

            // Add header parameters
            httpPost.setHeader("X-Auth-Token", token);

            // Add a body if you have specified the PUT or POST method. Special characters, such
            as the double quotation mark ("), contained in the body must be escaped.
            File file = new File(filePath);
            HttpEntity entity = MultipartEntityBuilder.create().addBinaryBody("images",
file).setContentType(ContentType.MULTIPART_FORM_DATA).setCharset(Consts.UTF_8).build();
            httpPost.setEntity(entity);

            // Send post
            CloseableHttpResponse response = HttpClients.createDefault().execute(httpPost);

            // Print result
            System.out.println(response.getStatusLine().getStatusCode());
            System.out.println(EntityUtils.toString(response.getEntity()));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

The **addBinaryBody** name is determined by the input parameter of the real-time service. The parameter name must be the same as that of the input parameter of the file type. The file **images** obtained in [Prerequisites](#) is used as an example.

– Text input (JSON)

The following is an example of the request body for reading the local inference file and performing Base64 encoding:

```
// Package name of the demo.
package com.apig.sdk.demo;

import org.apache.http.HttpHeaders;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;

public class MyTokenTest {

    public static void main(String[] args) {
        // Config url, token and body
        String url = "URL of the real-time service";
        String token = "User token";
```

```
String body = "{}";

try {
    // Create post
    HttpPost httpPost = new HttpPost(url);

    // Add header parameters
    httpPost.setHeader(HttpHeaders.CONTENT_TYPE, "application/json");
    httpPost.setHeader("X-Auth-Token", token);

    // Special characters, such as the double quotation mark ("), contained in the body
    must be escaped.
    httpPost.setEntity(new StringEntity(body));

    // Send post.
    CloseableHttpResponse response = HttpClient.createDefault().execute(httpPost);

    // Print result
    System.out.println(response.getStatusLine().getStatusCode());
    System.out.println(EntityUtils.toString(response.getEntity()));
} catch (Exception e) {
    e.printStackTrace();
}
```

body is determined by the text format. JSON is used as an example.

8.4.3.2 Accessing a Real-Time Service Through AK/SK-based Authentication

AK/SK-based authentication is a method used by ModelArts for identity authentication. Access Key (AK) and Secret Key (SK) are keys used to access Huawei Cloud services.

If a real-time service is in the **Running** state, it has been deployed. This service provides a standard, callable RESTful API. You can call the API using AK/SK-based authentication.

AK/SK-based authentication supports API requests with a body not larger than 12 MB. For API requests with a larger body, use token-based authentication. For details, see [Accessing a Real-Time Service Through Token-based Authentication](#).

When AK/SK-based authentication is used, you can use the APIG SDK or ModelArts SDK to access the real-time service. For details, see [Overview of Session Authentication](#). This section describes how to use the APIG SDK to access a real-time service. The process is as follows:

1. [Obtaining an AK/SK Pair](#)
2. [Obtaining Information About a Real-Time Service](#)
3. Send an inference request.
 - [Method 1: Use Python to Send an Inference Request](#)
 - [Method 2: Use Java to Send an Inference Request](#)

Constraints

- When you call an API to access a real-time service, the size of the prediction request body and the prediction time are subject to the following limitations:
 - The size of a request body cannot exceed 12 MB. Otherwise, the request will fail.

- Due to the limitation of API Gateway, the prediction duration of each request does not exceed 40 seconds.
- The local time on the client must be synchronized with the clock server to avoid a large offset in the value of the **X-Sdk-Date** request header. API Gateway checks the time format and compares the time with the time when API Gateway receives the request. If the time difference exceeds 15 minutes, API Gateway will reject the request.

Obtaining an AK/SK Pair

If an AK/SK pair is already available, skip this step. Find the downloaded AK/SK file, which is usually named **credentials.csv**.

The file contains the username, AK, and SK.

Figure 8-22 credential.csv

	A	B	C
1	User Name	Access Key Id	Secret Access Key
2	hu[REDACTED]dg	QTWA[REDACTED]UT2QVKYUC	MFyfvK41ba2[REDACTED]npdUKGpownRZImVmHc

To generate an AK/SK pair, follow these steps:

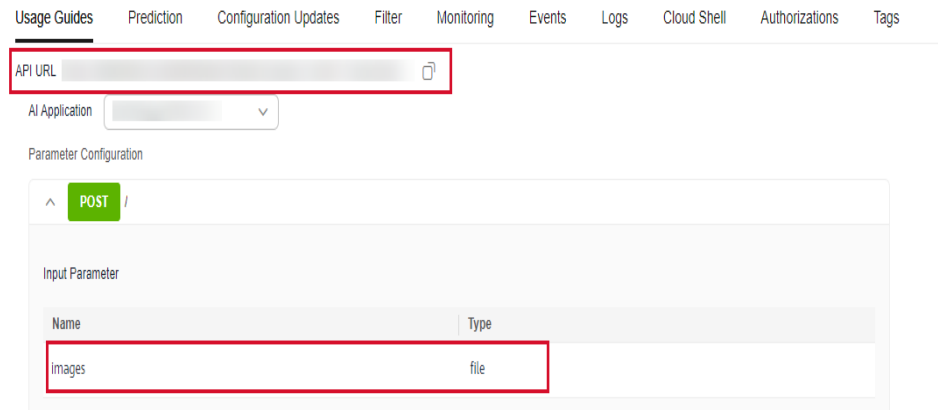
1. Register and log in to the [console](#).
2. Click the username and choose **My Credentials** from the drop-down list.
3. On the **My Credentials** page, choose **Access Keys** in the navigation pane.
4. Click **Create Access Key**.
5. Complete the identity authentication, download the access key, and keep it secure.

Obtaining Information About a Real-Time Service

To call an API, you will need the URL and input parameters of the real-time service. Follow these steps to obtain this information:

1. Log in to the [ModelArts console](#). In the navigation pane, choose **Model Deployment > Real-Time Services**.
2. Click the name of the target service to access its details page.
3. Obtain the URL and input parameters of the service.

The API URL is the service URL. If **apis** defines a path in the model configuration file, append the user-defined path to the URL, for example, *{URL of the real-time service}/predictions/poetry*.

Figure 8-23 Obtaining the API URL and file prediction input parameters of a real-time service

Method 1: Use Python to Send an Inference Request

1. Download the Python SDK and configure it in the development tool. For details, see [Integrating the Python SDK for API request signing](#).
2. Create a request body for inference.

– File input

```
# coding=utf-8

import requests
import os
from apig_sdk import signer

if __name__ == '__main__':
    # Config url, ak, sk and file path.
    url = "URL of the real-time service"
    # Hardcoded or plaintext AK/SK is risky. For security, encrypt your AK/SK and store
    # them in the configuration file or environment variables.
    # In this example, the AK/SK are stored in environment variables for identity
    # authentication. Before running this example, set environment variables
    # HUAWEICLOUD_SDK_AK and HUAWEICLOUD_SDK_SK.
    ak = os.environ["HUAWEICLOUD_SDK_AK"]
    sk = os.environ["HUAWEICLOUD_SDK_SK"]
    file_path = "Local path to the inference file"

    # Create request, set method, url, headers and body.
    method = 'POST'
    headers = {"x-sdk-content-sha256": "UNSIGNED-PAYLOAD"}
    request = signer.HttpRequest(method, url, headers)

    # Create sign, set the AK/SK to sign and authenticate the request.
    sig = signer.Signer()
    sig.Key = ak
    sig.Secret = sk
    sig.Sign(request)

    # Send request
    files = {'images': open(file_path, 'rb')}
    resp = requests.request(request.method, request.scheme + "://" + request.host + request.uri,
        headers=request.headers, files=files)

    # Print result
    print(resp.status_code)
    print(resp.text)
```

file_path is the local path to the inference file. The path can be an absolute path (for example, **D:/test.png** for Windows and **/opt/data/test.png** for Linux) or a relative path (for example, **./test.png**).

Request body format of **files**: `files = {"Request parameter": ("Load path", File content, "File type")}`. For details about parameters of **files**, see [Table 8-23](#).

Table 8-23 Parameters of **files**

Parameter	Mandatory	Description
Request parameter	Yes	Parameter name of the real-time service.
File path	No	Path for storing the file.
File content	Yes	Content of the file to be uploaded.
File type	No	Type of the file to be uploaded, which can be one of the following options: • txt: text/plain • jpg/jpeg: image/jpeg • png: image/png

– **Text input (JSON)**

The following is an example request body for reading the local inference file and performing Base64 encoding:

```
# coding=utf-8

import base64
import json
import os
import requests
from apig_sdk import signer

if __name__ == '__main__':
    # Config url, ak, sk and file path.
    url = "URL of the real-time service"
    # Hardcoded or plaintext AK/SK is risky. For security, encrypt your AK/SK and store them in the configuration file or environment variables.
    # In this example, the AK/SK are stored in environment variables for identity authentication. Before running this example, set environment variables HUAWEICLOUD_SDK_AK and HUAWEICLOUD_SDK_SK.
    ak = os.environ["HUAWEICLOUD_SDK_AK"]
    sk = os.environ["HUAWEICLOUD_SDK_SK"]
    file_path = "Local path to the inference file"
    with open(file_path, "rb") as file:
        base64_data = base64.b64encode(file.read()).decode("utf-8")

    # Create request, set method, url, headers and body.
    method = 'POST'
    headers = {
        'Content-Type': 'application/json'
    }
    body = {
        'image': base64_data
```

```
}
request = signer.HttpRequest(method, url, headers, json.dumps(body))

# Create sign, set the AK/SK to sign and authenticate the request.
sig = signer.Signer()
sig.Key = ak
sig.Secret = sk
sig.Sign(request)

# Send request
resp = requests.request(request.method, request.scheme + "://" + request.host + request.uri,
headers=request.headers, data=request.body)

# Print result
print(resp.status_code)
print(resp.text)
```

The **body** name is determined by the input parameter of the real-time service. The parameter name must be the same as that of the input parameter of the string type. **image** is used as an example. The value of **base64_data** in **body** is of the string type.

Method 2: Use Java to Send an Inference Request

1. Download the Java SDK and configure it in the development tool.
2. Create a Java request body for inference.

In the APIG Java SDK, **request.setBody()** can only be a string. Therefore, only text inference requests are supported. If a file is input, convert the file into text using Base64.

– File input

The following is an example request body (JSON) for reading the local inference file and performing Base64 encoding.

```
package com.apig.sdk.demo;
import com.cloud.apigateway.sdk.utils.Client;
import com.cloud.apigateway.sdk.utils.Request;
import org.apache.commons.codec.binary.Base64;
import org.apache.http.HttpHeaders;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.client.methods.HttpRequestBase;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;

public class MyAkSkTest2 {
    public static void main(String[] args) {
        String url = "URL of the real-time service";
        // Hard-coded or plaintext AK/SK is risky. For security, encrypt your AK/SK and store
        // them in the configuration file or environment variables.
        // In this example, the AK/SK are stored in environment variables for identity
        // authentication. Before running this example, set environment variables
        // HUAWEICLOUD_SDK_AK and HUAWEICLOUD_SDK_SK.
        String ak = System.getenv("HUAWEICLOUD_SDK_AK");
        String sk = System.getenv("HUAWEICLOUD_SDK_SK");
        String filePath = "Local path to the inference file";
        try {
            // Create request
            Request request = new Request();
            // Set the AK/SK to sign and authenticate the request.
            request.setKey(ak);
            request.setSecret(sk);
            // Specify a request method, such as GET, PUT, POST, DELETE, HEAD, and PATCH.
            request.setMethod(HttpPost.METHOD_NAME);
```

```
// Add header parameters
request.addHeader(HttpHeaders.CONTENT_TYPE, "application/json");
// Set a request URL in the format of https://{Endpoint}/{URI}.
request.setUrl(url);
// build your json body
String body = "{\"image\":\"\" + getBase64FromFile(filePath) + "\"}";
// Special characters, such as the double quotation mark ("), contained in the body
must be escaped.
request.setBody(body);
// Sign the request.
HttpRequestBase signedRequest = Client.sign(request);
// Send request.
CloseableHttpResponse response = HttpClients.createDefault().execute(signedRequest);
// Print result
System.out.println(response.getStatusLine().getStatusCode());
System.out.println(EntityUtils.toString(response.getEntity()));
} catch (Exception e) {
    e.printStackTrace();
}
}
/**
 * Convert the file into a byte array and Base64 encode it
 * @return
 */
private static String getBase64FromFile(String filePath) {
    // Convert the file into a byte array
    InputStream in = null;
    byte[] data = null;
    try {
        in = new FileInputStream(filePath);
        data = new byte[in.available()];
        in.read(data);
        in.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
    // Base64 encode
    return new String(Base64.encodeBase64(data));
}
```

 CAUTION

If using Base64 encoding, you need to add a decoding step to your model inference code to handle the request body.

– **Text input (JSON)**

```
// Package name of the demo.
package com.apig.sdk.demo;

import com.cloud.apigateway.sdk.utils.Client;
import com.cloud.apigateway.sdk.utils.Request;
import org.apache.http.HttpHeaders;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.client.methods.HttpRequestBase;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;

public class MyAkSkTest {

    public static void main(String[] args) {
        String url = "URL of the real-time service";
        // Hard-coded or plaintext AK/SK is risky. For security, encrypt your AK/SK and store
        them in the configuration file or environment variables.
        // In this example, the AK/SK are stored in environment variables for identity
```

authentication. Before running this example, set environment variables HUAWEICLOUD_SDK_AK and HUAWEICLOUD_SDK_SK.

```
String ak = System.getenv("HUAWEICLOUD_SDK_AK");
String sk = System.getenv("HUAWEICLOUD_SDK_SK");

try {
    // Create request
    Request request = new Request();

    // Set the AK/SK to sign and authenticate the request.
    request.setKey(ak);
    request.setSecret(sk);

    // Specify a request method, such as GET, PUT, POST, DELETE, HEAD, and PATCH.
    request.setMethod(HttpPost.METHOD_NAME);

    // Add header parameters
    request.addHeader(HttpHeaders.CONTENT_TYPE, "application/json");

    // Set a request URL in the format of https://{Endpoint}/{URI}.
    request.setUrl(url);

    // Special characters, such as the double quotation mark ("), contained in the body
    // must be escaped.
    String body = "{}";
    request.setBody(body);

    // Sign the request.
    HttpRequestBase signedRequest = Client.sign(request);

    // Send request.
    CloseableHttpResponse response = HttpClients.createDefault().execute(signedRequest);

    // Print result
    System.out.println(response.getStatusLine().getStatusCode());
    System.out.println(EntityUtils.toString(response.getEntity()));
} catch (Exception e) {
    e.printStackTrace();
}
```

body is determined by the text format. JSON is used as an example.

8.4.3.3 Accessing a Real-Time Service Through App Authentication

You can enable application authentication when deploying a real-time service. ModelArts registers an API that supports application authentication for the service. After this API is authorized to an application, you can call this API using the AppKey/AppSecret or AppCode of the application.

The process of application authentication for a real-time service is as follows:

1. **Enabling Application Authentication:** Enable application authentication. You can select or create an application.
2. **Managing Authorization for Real-Time Services:** Manage your applications, including viewing, resetting, or deleting them, as well as binding or unbinding real-time services and obtaining your AppKey and AppSecret, or AppCode.
3. **Application Authentication:** To call an API that supports app authentication, you will need to authenticate first. There are two authentication methods: AppKey and AppSecret, or AppCode. You can choose the one that suits you best.
4. Send an inference request.

- [Method 1: Use Python to Send an Inference request Through AppKey/AppSecret-based Authentication](#)
- [Method 2: Use Java to Send an Inference request Through AppKey/AppSecret-based Authentication](#)
- [Method 3: Use Python to Send an Inference request Through AppCode-based Authentication](#)
- [Method 4: Use Java to Send an Inference request Through AppCode-based Authentication](#)

Constraints

When you call an API to access a real-time service, the size of the prediction request body and the prediction time are subject to the following limitations:

- The size of a request body cannot exceed 12 MB. Otherwise, the request will fail.
- Due to the limitation of API Gateway, the prediction duration of each request does not exceed 40 seconds.

Prerequisites

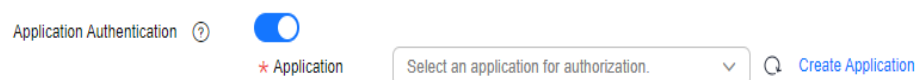
- A ModelArts model in the **Normal** state is available.
- The account is not in arrears to ensure available resources for service running.
- The local path to the inference file has been obtained. The path can be an absolute path (for example, **D:/test.png** for Windows and **/opt/data/test.png** for Linux) or a relative path (for example, **./test.png**).

Enabling Application Authentication

When deploying a real-time service, you can enable application authentication. You can also modify a deployed real-time service to support application authentication.

1. Log in to the ModelArts console and choose **Model Deployment > Real-Time Services**.
2. Enable application authentication.
 - When deploying a real-time service, enable application authentication on the **Deploy** page.
 - For a deployed real-time service, go to the **Real-Time Services** page, and click **Modify** in the **Operation** column of the service. On the service modification page, enable application authentication.

Figure 8-24 Enabling application authentication



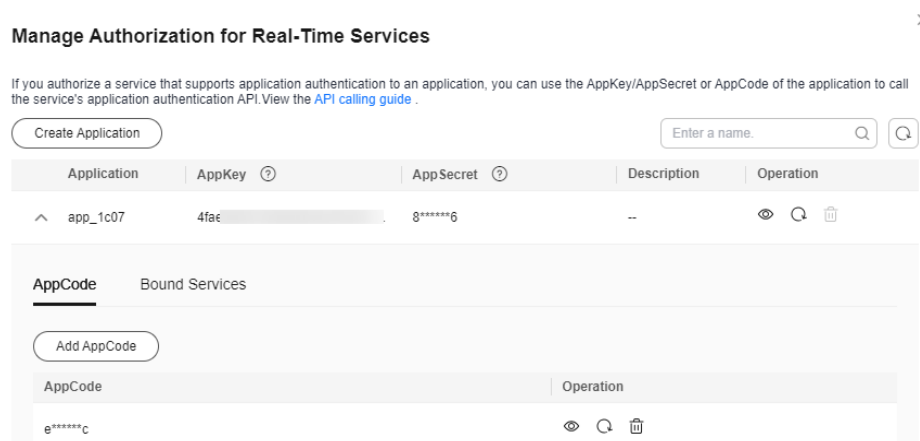
3. Select an application for authorization from the drop-down list. If no application is available, follow these steps to create one:

- Click **Create Application**, enter the application name and description, and click **OK**. By default, the application name is prefixed with **app_**. You can change this name if needed.
 - On the **Model Deployment > Real-Time Services** page, click **Authorize**. On the **Manage Authorization of Real-Time Services** page, click **Create Application**. For details, see [Managing Authorization for Real-Time Services](#).
4. After enabling application authentication, authorize a service that supports application authentication to the application. Then, you can use the created AppKey/AppSecret or AppCode to call the service's API that supports application authentication.

Managing Authorization for Real-Time Services

If you want to use application authentication, it is good practice to create an application on the authorization management page before deploying a real-time service. In the navigation pane, choose **Model Deployment > Real-Time Services**. On the **Real-Time Services** page, click **Authorize**. From there, you can create, reset, or delete applications, query plaintext, unbind real-time services from applications, and obtain the AppKey/AppSecret or AppCode.

Figure 8-25 Managing authorization for real-time services

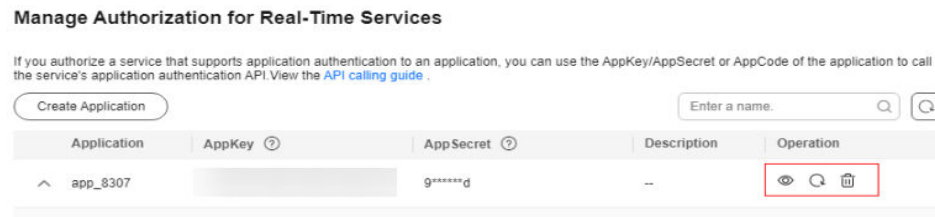


- **Creating an application**

Click **Create Application**, enter the application name and description, and click **OK**. By default, the application name is prefixed with **app_**. You can change this name if needed.

- **Viewing, resetting, or deleting an application**

Query plaintext, reset, or delete an application by clicking the corresponding icon in the **Operation** column of the application. After an application is created, the AppKey and AppSecret are automatically generated for application authentication.

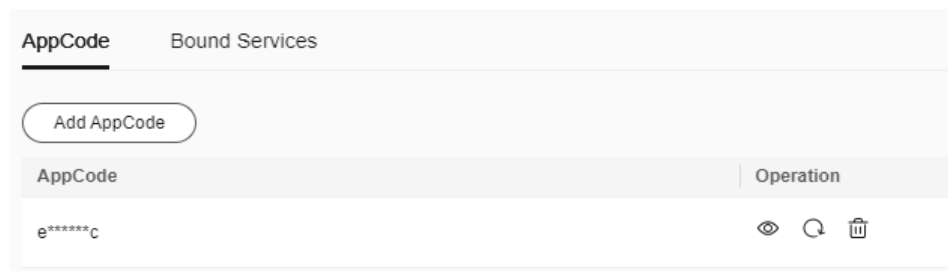
Figure 8-26 Query Plaintext, Reset, or Delete

- **Unbinding a service**

Click  next to the target application name to view the real-time services bound to the application. Click **Unbind** in the **Operation** column to cancel the binding. Then, this API cannot be called.

- **Obtaining the AppKey/AppSecret or AppCode**

Application authentication is required for API calling. The AppKey and AppSecret are automatically generated during application creation. Click  in the **Operation** column of the application in the **Manage Authorization of Real-Time Services** dialog box to view the complete AppSecret. Click  next to the application name to show the drop-down list. Click **Add AppCode** to automatically generate an AppCode. Then, click  in the **Operation** column to view the complete AppCode.

Figure 8-27 Adding the AppCode

Application Authentication

When a real-time service that supports application authentication is in the **Running** state, the service' API can be called. Before calling the API, perform application authentication.

When you use application authentication and enable simplified authentication, you can use your AppKey/AppSecret for signing and verification, or AppCode for simplified authentication. ModelArts uses simplified authentication by default. AppKey/AppSecret-based authentication is recommended because it is more secure than AppCode-based authentication.

- **AppKey/AppSecret-based authentication:** The AppKey and AppSecret are used to encrypt a request, identify the sender, and prevent the request from being modified. When using AppKey/AppSecret-based authentication, use a dedicated signing SDK to sign requests.
 - AppKey: access key ID of the application, which is a unique identifier used together with a secret access key to sign requests cryptographically.

- AppSecret: application secret access key, used together with the access key ID to encrypt the request, identify the sender, and prevent the request from being tempered.

AppKeys can be used for simplified authentication. When an API is called, the **apikey** parameter (value: **AppKey**) is added to the HTTP request header to accelerate authentication.

- **AppCode-based authentication:** Requests are authenticated using AppCodes. In AppCode-based authentication, the **X-Apig-AppCode** parameter (value: **AppCode**) is added to the HTTP request header when an API is called. The request content does not need to be signed. The API gateway only verifies the AppCode, achieving quick response.

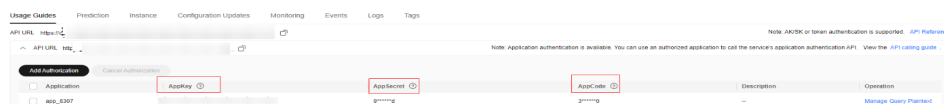
You can obtain the API URL (**url** of the real-time service), AppKey/AppSecret (**app_key** and **app_secret**), and AppCode (**app_code**) from the **Usage Guides** tab on the service details page (see [Figure 8-28](#)). Use the API URL for application authentication in the second line of the figure.

Modify the API URL in the following scenarios:

- If **apis** defines a path in the model configuration file, append the user-defined path to the URL, for example, *{URL of the real-time service}/predictions/poetry*.

If an SD WebUI inference service is deployed, add a slash (/) to the end of the calling address, for example, **https://8e*****5fe.apig.*****.huaweicloudapis.com/v1/infers/f2682*****f42/**.

Figure 8-28 Obtaining application authentication information



Method 1: Use Python to Send an Inference request Through AppKey/AppSecret-based Authentication

1. Download the Python SDK and configure it in the development tool.
2. Create a request body for inference.

- **File input**

```
# coding=utf-8

import requests
import os
from apig_sdk import signer

if __name__ == '__main__':
    # Config url, ak, sk and file path.
    # API URL, for example, "https://8e*****5fe.apig.*****.huaweicloudapis.com/v1/infers/f2682*****f42"
    url = "URL of the real-time service"
    # Hardcoded or plaintext app_key and app_secret are risky. For security, encrypt and store them in the configuration file or environment variables.
    # In this example, the app_key and app_secret are stored in environment variables for identity authentication. Before running this example, set environment variables HUAWEICLOUD_APP_KEY and HUAWEICLOUD_APP_SECRET.
    app_key = os.environ["HUAWEICLOUD_APP_KEY"]
    app_secret = os.environ["HUAWEICLOUD_APP_SECRET"]
    file_path = "Local path to the inference file"
```

```
# Create request, set method, url, headers and body.
method = 'POST'
headers = {"x-sdk-content-sha256": "UNSIGNED-PAYLOAD"}
request = signer.HttpRequest(method, url, headers)

# Create sign, set the AK/SK to sign and authenticate the request.
sig = signer.Signer()
sig.Key = app_key
sig.Secret = app_secret
sig.Sign(request)

# Send request
files = {'images': open(file_path, 'rb')}
resp = requests.request(request.method, request.scheme + "://" + request.host + request.uri,
headers=request.headers, files=files)

# Print result
print(resp.status_code)
print(resp.text)
```

Request body format of **files**: files = {"Request parameter": ("Load path", File content, "File type")}. For details about parameters of **files**, see [Table 8-24](#).

Table 8-24 Parameters of **files**

Parameter	Mandatory	Description
Request parameter	Yes	Parameter name of the real-time service.
File path	No	Path for storing the file.
File content	Yes	Content of the file to be uploaded.
File type	No	Type of the file to be uploaded, which can be one of the following options: • txt: text/plain • jpg/jpeg: image/jpeg • png: image/png

– **Text input (JSON)**

The following is an example request body for reading the local inference file and performing Base64 encoding:

```
# coding=utf-8

import base64
import json
import os
import requests
from apig_sdk import signer

if __name__ == '__main__':
    # Config url, ak, sk and file path.
    # API URL, for example, "https://8e*****5fe.apig.*****.huaweicloudapis.com/v1/infers/
    f2682*****f42"
```

```
url = "URL of the real-time service"
# Hardcoded or plaintext app_key and app_secret are risky. For security, encrypt and
store them in the configuration file or environment variables.
# In this example, the app_key and app_secret are stored in environment variables for
identity authentication. Before running this example, set environment variables
HUAWEICLOUD_APP_KEY and HUAWEICLOUD_APP_SECRET.
app_key = os.environ["HUAWEICLOUD_APP_KEY"]
app_secret = os.environ["HUAWEICLOUD_APP_SECRET"]
file_path = "Local path to the inference file"
with open(file_path, "rb") as file:
    base64_data = base64.b64encode(file.read()).decode("utf-8")

# Create request, set method, url, headers and body.
method = 'POST'
headers = {
    'Content-Type': 'application/json'
}
body = {
    'image': base64_data
}
request = signer.HttpRequest(method, url, headers, json.dumps(body))

# Create sign, set the AppKey&AppSecret to sign and authenticate the request.
sig = signer.Signer()
sig.Key = app_key
sig.Secret = app_secret
sig.Sign(request)

# Send request
resp = requests.request(request.method, request.scheme + "://" + request.host + request.uri,
headers=request.headers, data=request.body)

# Print result
print(resp.status_code)
print(resp.text)
```

The **body** name is determined by the input parameter of the real-time service. The parameter name must be the same as that of the input parameter of the string type. **image** is used as an example. The value of **base64_data** in **body** is of the string type.

Method 2: Use Java to Send an Inference request Through AppKey/ AppSecret-based Authentication

1. Download the Java SDK and configure it in the development tool.
2. Create a Java request body for inference.

In the APIG Java SDK, **request.setBody()** can only be a string. Therefore, only text inference requests are supported.

The following is an example of the request body (JSON) for reading the local inference file and performing Base64 encoding:

```
// Package name of the demo.
package com.apig.sdk.demo;

import com.cloud.apigateway.sdk.utils.Client;
import com.cloud.apigateway.sdk.utils.Request;
import org.apache.http.HttpHeaders;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.client.methods.HttpRequestBase;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;

public class MyAkSkTest {
```

```
public static void main(String[] args) {
    # API URL, for example, "https://8e*****5fe.apig.*****.huaweicloudapis.com/v1/infers/
    f2682*****f42"
    String url = "URL of the real-time service";
    // Hard-coded or plaintext app_key and app_secret are risky. For security, encrypt and store
    them in the configuration file or environment variables.
    // In this example, the app_key and app_secret are stored in environment variables for
    identity authentication. Before running this example, set environment variables
    HUAWEICLOUD_APP_KEY and HUAWEICLOUD_APP_SECRET.
    String appKey = System.getenv("HUAWEICLOUD_APP_KEY");
    String appSecret = System.getenv("HUAWEICLOUD_APP_SECRET");
    String body = "{}";

    try {
        // Create request
        Request request = new Request();

        // Set the AK/AppSecret to sign and authenticate the request.
        request.setKey(appKey);
        request.setSecret(appSecret);

        // Specify a request method, such as GET, PUT, POST, DELETE, HEAD, and PATCH.
        request.setMethod(HttpPost.METHOD_NAME);

        // Add header parameters
        request.addHeader(HttpHeaders.CONTENT_TYPE, "application/json");

        // Set a request URL in the format of https://{Endpoint}/{URI}.
        request.setUrl(url);

        // Special characters, such as the double quotation mark ("), contained in the body must be
        escaped.
        request.setBody(body);

        // Sign the request.
        HttpRequestBase signedRequest = Client.sign(request);

        // Send request.
        CloseableHttpResponse response = HttpClient.createDefault().execute(signedRequest);

        // Print result
        System.out.println(response.getStatusLine().getStatusCode());
        System.out.println(EntityUtils.toString(response.getEntity()));
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

body is determined by the text format. JSON is used as an example.

Method 3: Use Python to Send an Inference request Through AppCode-based Authentication

1. Download the Python SDK and configure it in the development tool.
2. Create a request body for inference.

– **File input**

```
# coding=utf-8

import requests
import os

if __name__ == '__main__':
    # Config url, app code and file path.
    # API URL, for example, "https://8e*****5fe.apig.*****.huaweicloudapis.com/v1/infers/
    f2682*****f42"
    url = "URL of the real-time service"
```

```
# Hardcoded or plaintext app_code is risky. For security, encrypt and store it in the
configuration file or environment variables.
# In this example, the app_code is stored in environment variables for identity
authentication. Before running this example, set environment variable
HUAWEICLOUD_APP_CODE.
app_code = os.environ["HUAWEICLOUD_APP_CODE"]
file_path = "Local path to the inference file"

# Send request.
headers = {
    'X-Apig-AppCode': app_code
}
files = {
    'images': open(file_path, 'rb')
}
resp = requests.post(url, headers=headers, files=files)

# Print result
print(resp.status_code)
print(resp.text)
```

The **files** name is determined by the input parameter of the real-time service. The parameter name must be the same as that of the input parameter of the file type. In this example, **images** is used.

– Text input (JSON)

The following is an example request body for reading the local inference file and performing Base64 encoding:

```
# coding=utf-8

import base64
import requests
import os

if __name__ == '__main__':
    # Config url, app code and request body.
    # API URL, for example, "https://8e*****5fe.apig.*****.huaweicloudapis.com/v1/infers/
    f2682*****f42"
    url = "URL of the real-time service"
    # Hardcoded or plaintext app_code is risky. For security, encrypt and store it in the
    configuration file or environment variables.
    # In this example, the app_code is stored in environment variables for identity
    authentication. Before running this example, set environment variable
    HUAWEICLOUD_APP_CODE.
    app_code = os.environ["HUAWEICLOUD_APP_CODE"]
    file_path = "Local path to the inference file"
    with open(file_path, "rb") as file:
        base64_data = base64.b64encode(file.read()).decode("utf-8")

    # Send request
    headers = {
        'Content-Type': 'application/json',
        'X-Apig-AppCode': app_code
    }
    body = {
        'image': base64_data
    }
    resp = requests.post(url, headers=headers, json=body)

    # Print result
    print(resp.status_code)
    print(resp.text)
```

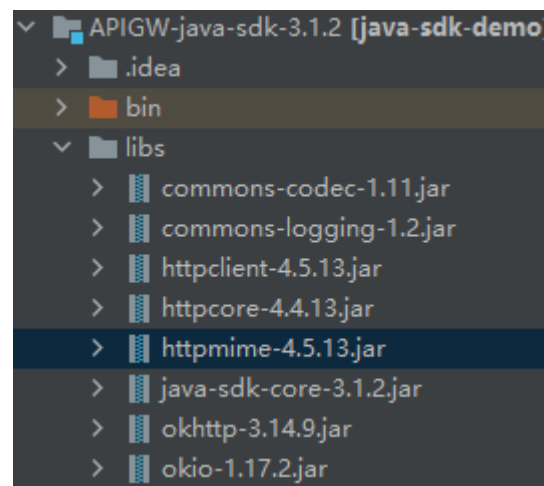
The **body** name is determined by the input parameter of the real-time service. The parameter name must be the same as that of the input parameter of the string type. **image** is used as an example. The value of **base64_data** in **body** is of the string type.

Method 4: Use Java to Send an Inference request Through AppCode-based Authentication

1. Download the Java SDK and configure it in the development tool.
2. (Optional) If the inference request input is in a file format, follow these steps to ensure the Java project includes the httpmime module as a dependency.
 - a. Add **httpmime-x.x.x.jar** to the **libs** folder. **Figure 8-29** shows a complete Java dependency library.

You are advised to use httpmime-x.x.x.jar 4.5 or later. Download httpmime-x.x.x.jar from <https://mvnrepository.com/artifact/org.apache.httpcomponents/httpmime>.

Figure 8-29 Java dependency library



- b. After **httpmime-x.x.x.jar** is added, add httpmime information to the **.classpath** file of the Java project as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<classpath>
<classpathentry kind="con" path="org.eclipse.jdt.launching.JRE_CONTAINER"/>
<classpathentry kind="src" path="src"/>
<classpathentry kind="lib" path="libs/commons-codec-1.11.jar"/>
<classpathentry kind="lib" path="libs/commons-logging-1.2.jar"/>
<classpathentry kind="lib" path="libs/httpclient-4.5.13.jar"/>
<classpathentry kind="lib" path="libs/httpcore-4.4.13.jar"/>
<classpathentry kind="lib" path="libs/httpmime-x.x.x.jar"/>
<classpathentry kind="lib" path="libs/java-sdk-core-3.1.2.jar"/>
<classpathentry kind="lib" path="libs/okhttp-3.14.9.jar"/>
<classpathentry kind="lib" path="libs/okio-1.17.2.jar"/>
<classpathentry kind="output" path="bin"/>
</classpath>
```

3. Create a Java request body for inference.
 - **File input**

A sample Java request body is as follows:

```
// Package name of the demo.
package com.apigw.sdk.demo;

import org.apache.http.Consts;
import org.apache.http.HttpEntity;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.ContentType;
import org.apache.http.entity.mime.MultipartEntityBuilder;
```

```
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;

import java.io.File;

public class MyAppCodeFile {

    public static void main(String[] args) {
        # API URL, for example, "https://8e*****5fe.apig.*****.huaweicloudapis.com/v1/infers/
f2682*****f42"
        String url = "URL of the real-time service";
        // Hard-coded or plaintext appCode is risky. For security, encrypt and store it in the
configuration file or environment variables.
        // In this example, the appCode is stored in environment variables for identity
authentication. Before running this example, set environment variable
HUAWEICLOUD_APP_CODE.
        String appCode = System.getenv("HUAWEICLOUD_APP_CODE");
        String filePath = "Local path to the inference file";

        try {
            // Create post
            HttpPost httpPost = new HttpPost(url);

            // Add header parameters
            httpPost.setHeader("X-Apig-AppCode", appCode);

            // Special characters, such as the double quotation mark ("), contained in the body
must be escaped.
            File file = new File(filePath);
            HttpEntity entity = MultipartEntityBuilder.create().addBinaryBody("images",
file).setContentType(ContentType.MULTIPART_FORM_DATA).setCharset(Consts.UTF_8).build();
            httpPost.setEntity(entity);

            // Send post
            CloseableHttpResponse response = HttpClients.createDefault().execute(httpPost);

            // Print result
            System.out.println(response.getStatusLine().getStatusCode());
            System.out.println(EntityUtils.toString(response.getEntity()));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

The **addBinaryBody** name is determined by the input parameter of the real-time service. The parameter name must be the same as that of the input parameter of the file type. In this example, **images** is used.

– **Text input (JSON)**

The following is an example request body for reading the local inference file and performing Base64 encoding:

```
// Package name of the demo.
package com.apig.sdk.demo;

import org.apache.http.HttpHeaders;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;

public class MyAppCodeTest {

    public static void main(String[] args) {
        # API URL, for example, "https://8e*****5fe.apig.*****.huaweicloudapis.com/v1/infers/
f2682*****f42"
        String url = "URL of the real-time service";
```



```
// Hard-coded or plaintext appCode is risky. For security, encrypt and store it in the
// configuration file or environment variables.
// In this example, the appCode is stored in environment variables for identity
// authentication. Before running this example, set environment variable
// HUAWEICLOUD_APP_CODE.
String appCode = System.getenv("HUAWEICLOUD_APP_CODE");
String body = "{}";

try {
    // Create post
    HttpPost httpPost = new HttpPost(url);

    // Add header parameters
    httpPost.setHeader(HttpHeaders.CONTENT_TYPE, "application/json");
    httpPost.setHeader("X-Apig-AppCode", appCode);

    // Special characters, such as the double quotation mark ("), contained in the body
    // must be escaped.
    httpPost.setEntity(new StringEntity(body));

    // Send post
    CloseableHttpResponse response = HttpClient.createDefault().execute(httpPost);

    // Print result
    System.out.println(response.getStatusLine().getStatusCode());
    System.out.println(EntityUtils.toString(response.getEntity()));
} catch (Exception e) {
    e.printStackTrace();
}
```

body is determined by the text format. JSON is used as an example.

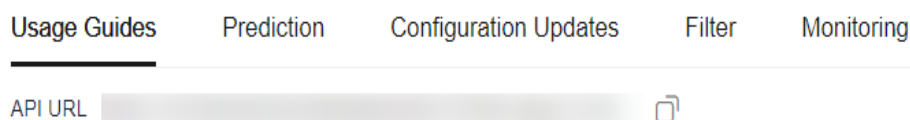
8.4.4 Accessing a Real-Time Service Through Different Channels

8.4.4.1 Accessing a Real-Time Service Through a Public Network

Context

ModelArts inference accesses real-time services through the public network by default, supporting both HTTPS and WebSocket protocols. After a real-time service is deployed, a standard RESTful API is provided for you to call. You can view the API URL on the **Usage Guides** tab page of the service details page.

Figure 8-30 API URL



Constraints

When you call an API to access a real-time service, the size of the prediction request body and the prediction time are subject to the following limitations:

- The size of a request body cannot exceed 12 MB. Otherwise, the request will fail.
- Due to the limitation of API Gateway, the prediction duration of each request does not exceed 40 seconds.

Accessing a Real-Time Service

The following authentication modes are available for accessing real-time services from a public network:

- [Accessing a Real-Time Service Through Token-based Authentication](#)
- [Accessing a Real-Time Service Through AK/SK-based Authentication](#)
- [Accessing a Real-Time Service Through App Authentication](#)

8.4.4.2 Accessing a Real-Time Service Through a VPC Channel

Context

To access a ModelArts real-time service from an internal VPC node of your account, you can use a VPC channel. By creating an endpoint in your VPC and connecting to the ModelArts VPC endpoint service, you can access the real-time service from your VPC endpoint. VPC access channels work best when you need strong security and fast connections, like for internal system communications.

Constraints

When you call an API to access a real-time service, the size of the prediction request body and the prediction time are subject to the following limitations:

- The size of a request body cannot exceed 12 MB. Otherwise, the request will fail.
- Due to the limitation of API Gateway, the prediction duration of each request does not exceed 40 seconds.

Procedure

To access a real-time service through a VPC channel, follow these steps:

1. [Obtain the ModelArts VPC endpoint service address.](#)
2. [Buy and connect to a ModelArts endpoint.](#)
3. [Create a private DNS zone.](#)
4. [Access a real-time service through VPC.](#)

Step 1 Submit a service ticket and provide the account ID to Huawei Cloud technical support to obtain the ModelArts VPC endpoint service address.

Step 2 Buy and connect to a ModelArts endpoint.

1. Log in to the VPC management console. In the navigation pane, choose **VPC Endpoint > VPC Endpoints**.
2. Click **Buy VPC Endpoint** in the upper right corner.

- **Region:** region where the VPC endpoint is located.
Resources in different regions cannot communicate with each other. The region must be the same as that of ModelArts.
 - **Service Category:** Select **Find a service by name**.
 - **VPC Endpoint Service Name:** Enter the endpoint service address obtained in [step 1](#). Click **Verify** on the right. The system automatically sets **VPC**, **Subnet**, and **Private IP Address**.
 - **Create a Private Domain Name:** Retain the default setting.
3. Confirm the specifications, and click **Next** and then **Submit**. The VPC endpoint list page is displayed.

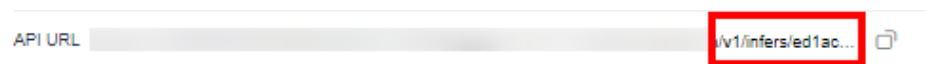
Step 3 Create a private DNS zone.

The newly created real-time service is interconnected with a dedicated gateway. The independent public domain name of ModelArts inference, that is, **infer-modelarts-*<Region ID>*.modelarts-infer.com**, is required. The intranet VPC cannot resolve the **modelarts-infer.com** domain name. You need to add private domain name resolution by referring to this step and [Step 4](#).

1. Log in to the DNS console. In the navigation pane on the left, choose **Private Zones**.
2. Click **Create Private Zone**. Set the following parameters:
 - **Domain Name:** Enter a value in the format **infer-modelarts-*<Region ID>*.modelarts-infer.com**. Example: **infer-modelarts-cn-south-1.modelarts-infer.com**.
 - **VPC:** Select a VPC you want to associate with the private zone.
3. Click **OK**.

Step 4 Access the real-time service through the VPC.

1. Use the following API to access a real-time service through VPC:
`https://{Private DNS domain name}/{URL}`
 - *Private DNS domain name*: private domain name you set. You can also click **Access VPC** on the real-time service list page to view the domain name in the displayed dialog box.
 - *URL*: The URL for a real-time service is the part after the domain name of **API URL** in the **Usage Guides** tab of the service details page.

Figure 8-31 Obtaining the URL

2. Use GUI-based software, cURL command, or Python to access a real-time service. For details, see [Accessing a Real-Time Service Through Token-based Authentication](#).

----End

8.4.4.3 Accessing a Real-Time Service Through a VPC High-Speed Channel

Context

When accessing a real-time service, you may require:

- High throughput and low latency
- TCP or RPC requests

To meet these requirements, ModelArts enables high-speed access through VPC peering.

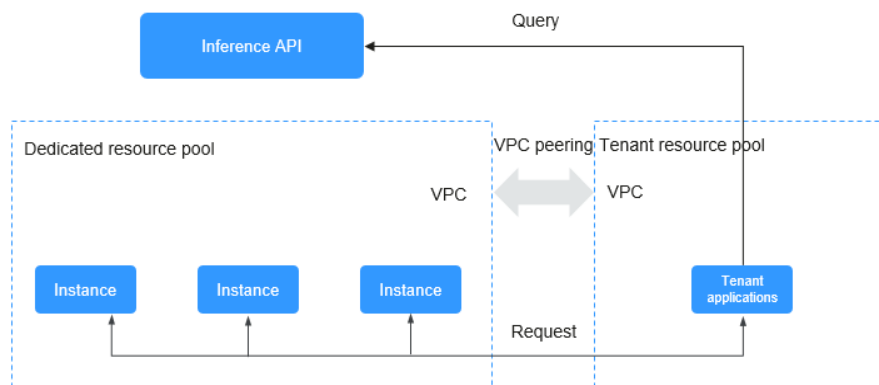
In high-speed access through VPC peering, your service requests are directly sent to instances through VPC peering but not through the inference platform. This accelerates service access. This method works best when you need high bandwidth with minimal delays, like handling real-time data or video streams.

WARNING

When using a high-speed VPC access channel for a real-time service, requests do not reach the inference platform, making these features inaccessible:

- Authentication
 - Traffic distribution by configuration
 - Load balancing
 - Alarm, monitoring, and statistics
-

Figure 8-32 High-speed access through VPC peering



Constraints

When you call an API to access a real-time service, the size of the prediction request body and the prediction time are subject to the following limitations:

- The size of a request body cannot exceed 12 MB. Otherwise, the request will fail.
- Due to the limitation of API Gateway, the prediction duration of each request does not exceed 40 seconds.

- Only the services deployed in a dedicated resource pool support high-speed access through VPC peering.
- High-speed access through VPC peering is available only for real-time services.
- Due to traffic control, there is a limit on how often you can get the IP address and port number of a real-time service. The number of calls of each tenant account cannot exceed 2000 per minute, and that of each IAM user account cannot exceed 20 per minute.
- High-speed access through VPC peering is available only for the services deployed using the AI applications imported from custom images.

Preparations

Deploy a real-time service in a dedicated resource pool and ensure the service is running.

Procedure

To enable high-speed access to a real-time service through VPC peering, perform the following operations:

1. **Interconnect the dedicated resource pool to the VPC.**
2. **Create an ECS in the VPC.**
3. **Obtain the IP address and port number of the real-time service.**
4. **Access the service through the IP address and port number.**

Step 1 Interconnect the dedicated resource pool to the VPC.

Log in to the ModelArts console, choose **Resource Management** > **Standard Cluster**, find the dedicated resource pool where the service is deployed, and click its name/ID to go to the resource pool details page. Obtain the network configuration. Return to the homepage, choose **Network** in the navigation pane, find the network associated with the dedicated resource pool, and interconnect your VPC. After the VPC is accessed, the VPC will be displayed on the network list and resource pool details pages. Click the VPC to go to the details page.

Figure 8-33 Obtaining the network configuration



Figure 8-34 Interconnecting the VPC



Step 2 Create an ECS in the VPC.

Log in to the ECS management console and click **Buy ECS** in the upper right corner. On the **Buy ECS** page, configure basic settings and click **Next: Configure Network**. On the **Configure Network** page, select the VPC connected in [Step 1](#), configure other parameters, confirm the settings, and click **Submit**. When the ECS status changes to **Running**, the ECS has been created. Click its name/ID to go to the server details page and view the VPC configuration.

Figure 8-35 Selecting a VPC when purchasing an ECS

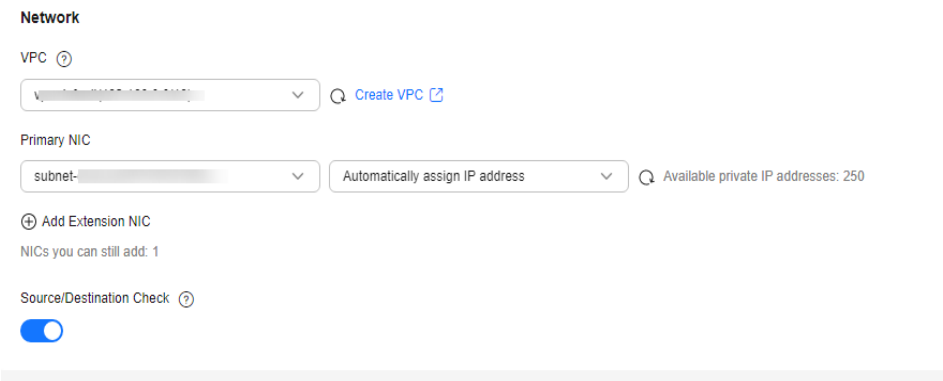
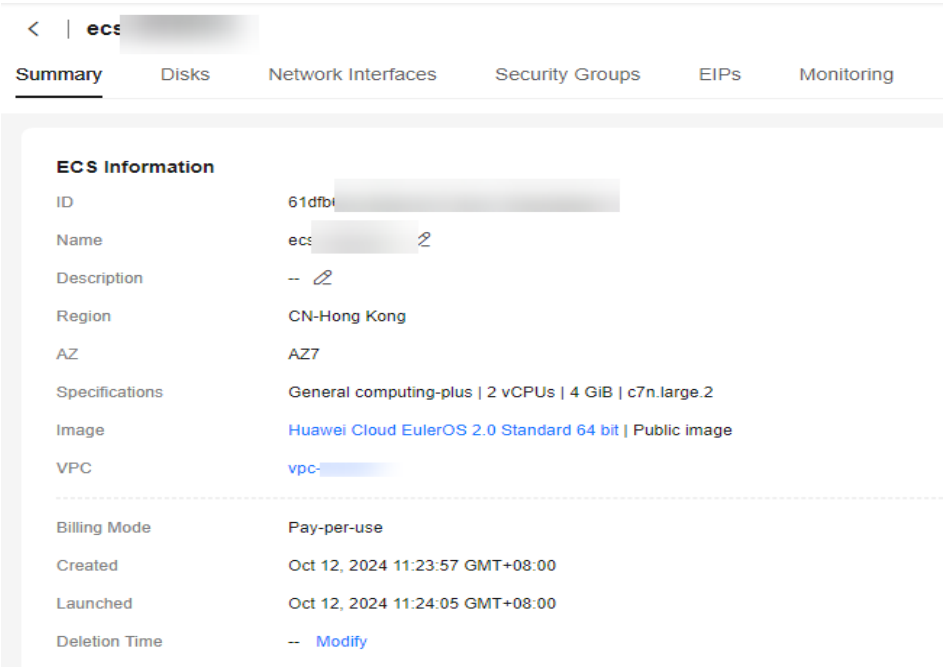


Figure 8-36 Viewing VPC information



Step 3 Obtain the IP address and port number of the real-time service.

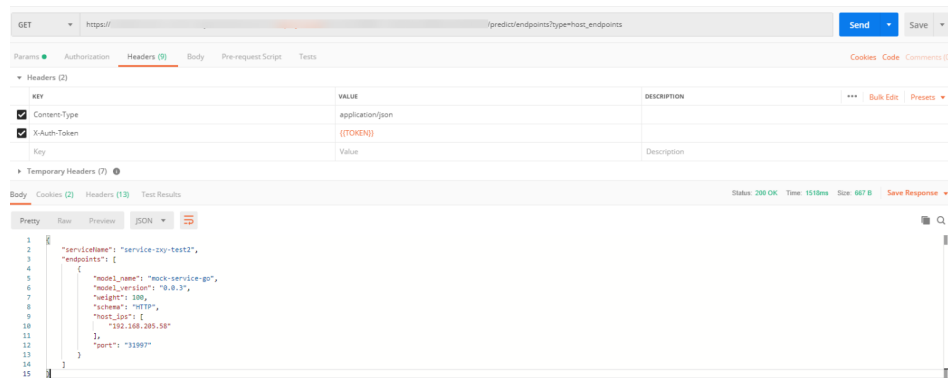
GUI software, for example, Postman can be used to obtain the IP address and port number. Alternatively, log in to the ECS, create a Python environment, and execute code to obtain the service IP address and port number.

API:

GET /v1/{project_id}/services/{service_id}/predict/endpoints?type=host_endpoints

- Method 1: Obtain the IP address and port number using GUI software.

Figure 8-37 Example response



- Method 2: Obtain the IP address and port number using Python.
The following parameters in the Python code below need to be modified:
- **project_id**: your project ID. To obtain it, see [Obtaining a Project ID and Name](#).
- **service_id**: service ID, which can be viewed on the service details page.
- **REGION_ENDPOINT**: service endpoint. To obtain it, see [Endpoint](#).

```
def get_app_info(project_id, service_id):
    list_host_endpoints_url = "{}/v1/{}/services/{}/predict/endpoints?type=host_endpoints"
    url = list_host_endpoints_url.format(REGION_ENDPOINT, project_id, service_id)
    headers = {'X-Auth-Token': X_Auth-Token}
    response = requests.get(url, headers=headers)
    print(response.content)
```

Step 4 Access the service through the IP address and port number.

Log in to the ECS and access the real-time service either by running Linux commands or by creating a Python environment and executing Python code. Obtain the values of **schema**, **ip**, and **port** from [Step 3](#).

- Run the following command to access the real-time service:

```
curl --location --request POST 'http://192.168.205.58:31997' \
--header 'Content-Type: application/json' \
--data-raw '{"a":"a"}'
```

Figure 8-38 Accessing a real-time service

```
[root@ecs-zxy ~]# curl --location --request POST 'http://192.168.205.58:31997' \
> --header 'Content-Type: application/json' \
> --data-raw '{"a":"a"}'
call Post()[root@ecs-zxy ~]# _
```

- Create a Python environment and execute Python code to access the real-time service.

```
def vpc_infer(schema, ip, port, body):
    infer_url = "{}/://{}/{}"
    url = infer_url.format(schema, ip, port)
    response = requests.post(url, data=body)
    print(response.content)
```

 NOTE

High-speed access does not support load balancing. You need to customize load balancing policies when you deploy multiple instances.

-----End

8.4.5 Accessing a Real-Time Service Using Different Protocols

8.4.5.1 Accessing a Real-Time Service Using WebSocket

Context

WebSocket is a network transmission protocol that supports full-duplex communication over a single TCP connection. It is located at the application layer in the OSI model. The WebSocket communication protocol was established by IETF as standard RFC 6455 in 2011 and supplemented by RFC 7936. The WebSocket API in the Web IDL is standardized by W3C.

WebSocket simplifies data exchange between the client and server and allows the server to proactively push data to the client. In the WebSocket API, if the initial handshake between the client and server is successful, a persistent connection can be established between them and bidirectional data transmission can be performed. WebSocket is ideal for applications requiring real-time, bidirectional communication, such as online games and real-time chat.

WebSocket can be used to deploy real-time services on ModelArts, particularly when custom images are needed to import AI models. WebSocket only supports real-time services and, when used on ModelArts, the protocol is converted to WebSocket Secure (WSS), which supports one-way authentication.

Key features of WebSocket include:

- **Persistent connection:** Unlike HTTP, WebSocket maintains a persistent connection between the client and server, reducing the overhead of frequent connection establishment.
- **Bidirectional communication:** Data can flow in both directions, allowing the server to proactively send data to the client, not just respond to client requests.
- **Reduced latency and bandwidth:** Real-time data transmission is possible because the connection remains open, minimizing latency and reducing unnecessary HTTP requests and bandwidth usage.

Prerequisites

- A real-time service has been deployed with **WebSocket** enabled.
- The image for importing the model is WebSocket-compliant.

Constraints

- WebSocket supports only the deployment of real-time services.
- It supports only real-time services deployed using models imported from custom images.

- When you call an API to access a real-time service, the size of the prediction request body and the prediction time are subject to the following limitations:
 - The size of a request body cannot exceed 12 MB. Otherwise, the request will fail.
 - Due to the limitation of API Gateway, the prediction duration of each request does not exceed 40 seconds.

Calling a WebSocket Real-Time Service

WebSocket itself does not require additional authentication. ModelArts WebSocket is WebSocket Secure-compliant, regardless of whether WebSocket or WebSocket Secure is enabled in the custom image. WebSocket Secure supports only one-way authentication, from the client to the server.

You can use one of the following authentication methods provided by ModelArts:

- **Token-based Authentication**
- **AK/SK**
- **App Authentication**

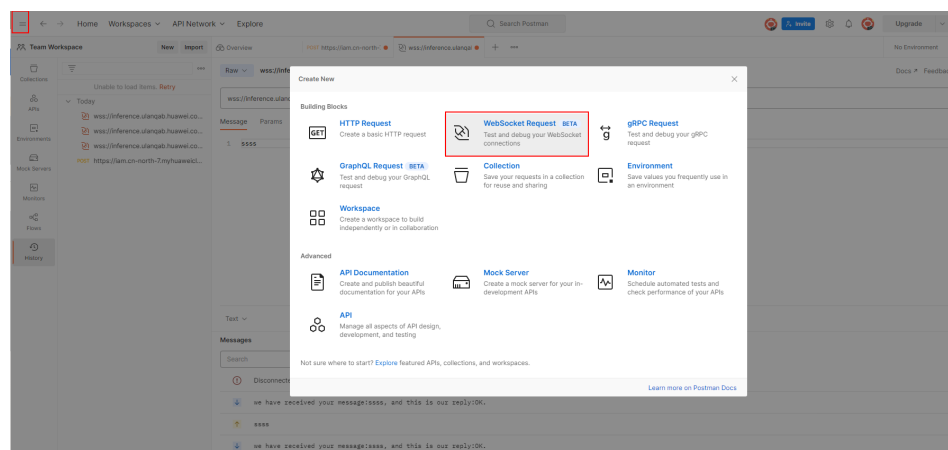
The following section uses GUI software Postman for prediction and token authentication as an example to describe how to call WebSocket.

1. **Establish a WebSocket connection.**
2. **Exchange data between the WebSocket client and the server.**

Step 1 Establish a WebSocket connection.

1. Open Postman of a version later than 8.5, for example, 10.12.0. Click  in the upper left corner and choose **File > New**. In the displayed dialog box, select **WebSocket Request** (beta version currently).

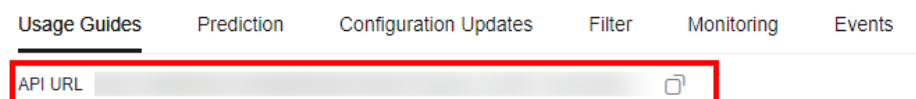
Figure 8-39 WebSocket Request



2. Configure parameters for the WebSocket connection.
Select **Raw** in the upper left corner. Do not select **Socket.IO** (a type of WebSocket implementation, which requires that both the client and the server run on **Socket.IO**). In the address box, enter the **API Address** obtained on the **Usage Guides** tab on the service details page. If there is a finer-grained URL

in the custom image, add the URL to the end of the address. If **queryString** is available, add this parameter in the **params** column. Add authentication information into the header. The header varies depending on the authentication mode, which is the same as that in the HTTPS-compliant inference service. Click **Connect** in the upper right corner to establish a WebSocket connection.

Figure 8-40 Obtaining the API address

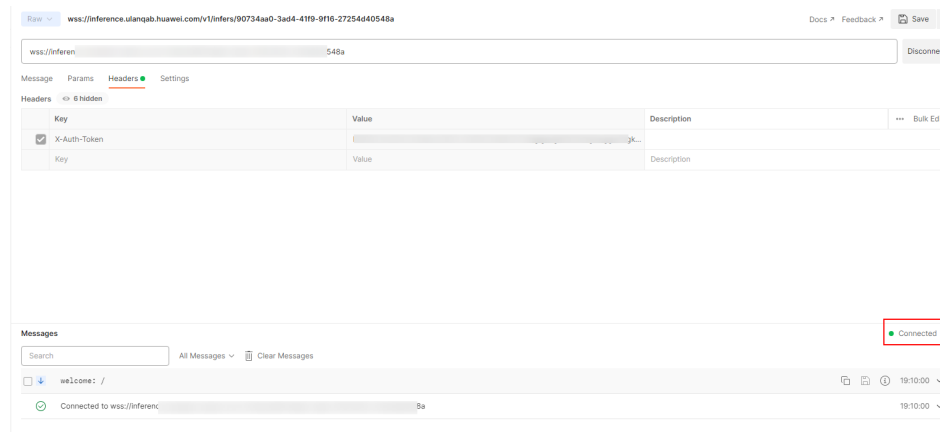


NOTE

- If the information is correct, **CONNECTED** will be displayed in the lower right corner.
- If establishing the connection failed and the status code is 401, check the authentication.
- If a keyword such as **WRONG_VERSION_NUMBER** is displayed, check whether the port configured in the custom image is the same as that configured in WebSocket or WebSocket Secure.

The following shows an established WebSocket connection.

Figure 8-41 Connection established



NOTICE

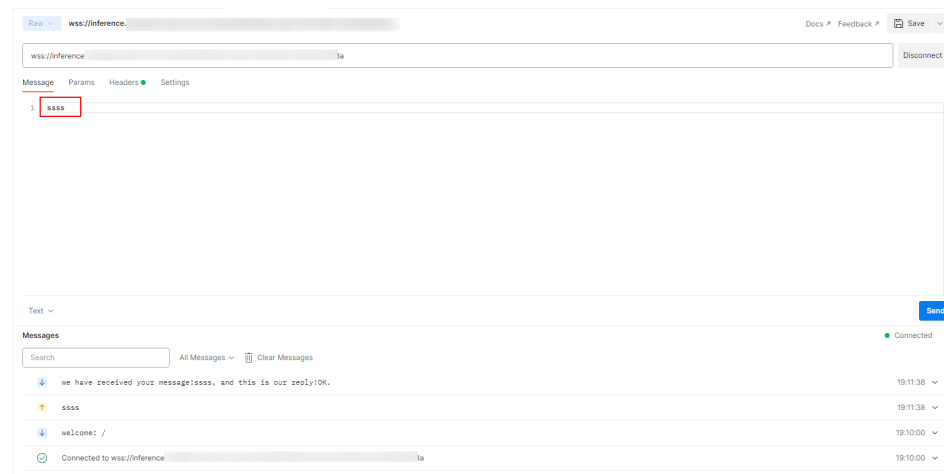
Preferentially check the WebSocket service provided by the custom image. The type of implementing WebSocket varies depending on the tool you used. Possible issues are as follows: A WebSocket connection can be established but cannot be maintained, or the connection is interrupted after one request and needs to be reconnected. ModelArts only ensures that it will not affect the WebSocket status in a custom image (the API address and authentication mode may be changed on ModelArts).

Step 2 Exchange data between the WebSocket client and the server.

After the connection is established, WebSocket uses TCP for full-duplex communication. The WebSocket client sends data to the server. The implementation types vary depending on the client, and the lib package may also be different for the same language. Different implementation types are not considered here.

The format of the data sent by the client is not limited by the protocol. Postman supports text, JSON, XML, HTML, and Binary data. Take text as an example. Enter the text data in the text box and click **Send** on the right to send the request to the server. If the text is oversized, Postman may be suspended.

Figure 8-42 Sending data



----End

8.4.5.2 Accessing a Real-Time Service Using Server-Sent Events

Context

Server-Sent Events (SSE) is a server push technology enabling a server to push events to a client via an HTTP connection. This technology is usually used to enable a server to unidirectionally push real-time data to a client, for example, a real-time news update or stock prices.

SSE primarily facilitates unidirectional real-time communication from the server to the client, such as streaming ChatGPT responses. In contrast to WebSockets, which provide bidirectional real-time communication, SSE is designed to be more lightweight and simpler to implement.

Key features of SSE include:

- **Easy to use:** SSE is based on the HTTP protocol and is straightforward to implement. No complex configurations or additional libraries are needed, and data can be pushed in real-time over standard HTTP connections.
- **Automatic reconnection:** SSE supports automatic reconnection. If the connection is interrupted, the client automatically attempts to reconnect, ensuring continuous data delivery.

- Unidirectional communication: SSE is unidirectional, meaning the server can send events to the client, but the client cannot send data back to the server through the same connection.
- Low resource usage: SSE uses HTTP connections, making it less resource-intensive compared to other real-time communication protocols like WebSocket. This makes SSE ideal for lightweight real-time data push scenarios.

Prerequisites

The image for importing the model is SSE-compliant.

Constraints

- SSE supports only the deployment of real-time services.
- It supports only real-time services deployed using models imported from custom images.
- When you call an API to access a real-time service, the size of the prediction request body and the prediction time are subject to the following limitations:
 - The size of a request body cannot exceed 12 MB. Otherwise, the request will fail.
 - Due to the limitation of API Gateway, the prediction duration of each request does not exceed 40 seconds.

Calling an SSE Real-Time Service

The SSE protocol itself does not introduce new authentication mechanisms; it relies on the same methods as HTTP requests.

You can use one of the following authentication methods provided by ModelArts:

- [Accessing a Real-Time Service Through Token-based Authentication](#)
- [Accessing a Real-Time Service Through AK/SK-based Authentication](#)
- [Accessing a Real-Time Service Through App Authentication](#)

The following section uses GUI software Postman for prediction and token authentication as an example to describe how to call an SSE service.

Figure 8-43 Calling an SSE service

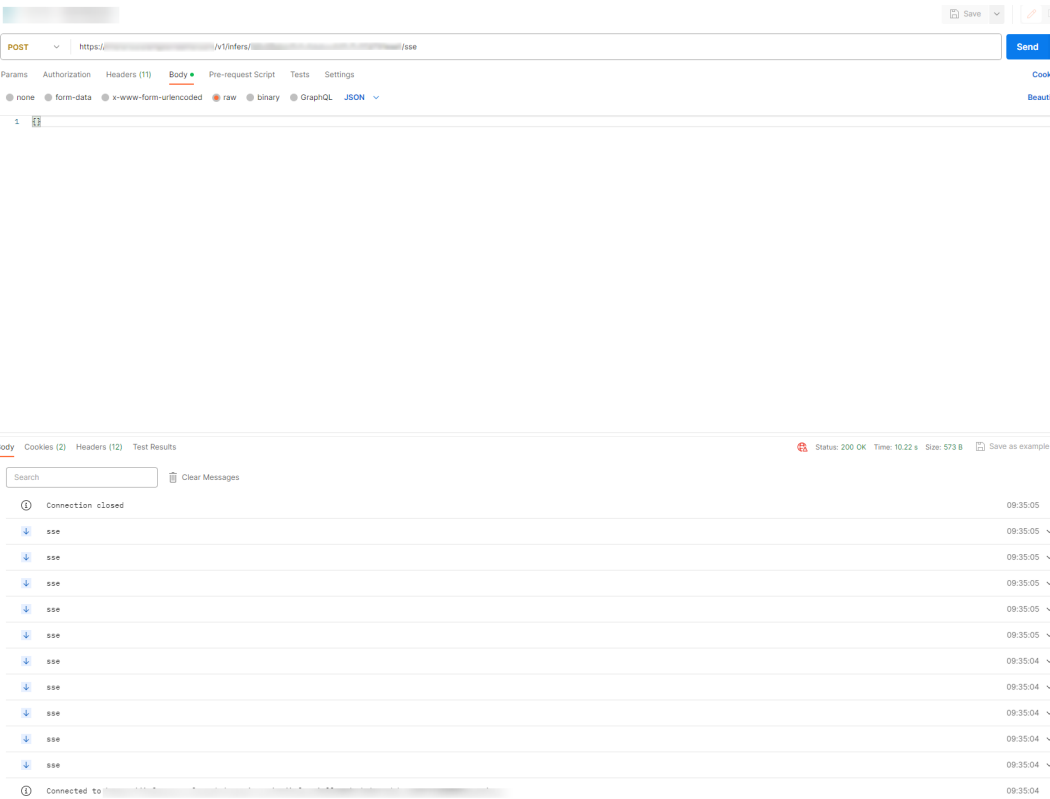


Figure 8-44 Response header Content-Type

Body Cookies Headers Test Results	
KEY	VALUE
Content-Type ⓘ	text/event-stream;charset=UTF-8

NOTE

In normal cases, the value of **Content-Type** in the response header is **text/event-stream;charset=UTF-8**.

8.5 Deploying a Model as a Batch Inference Service

Batch inference: Processes multiple inputs at once and returns all results together. ModelArts allows you to deploy a model as a batch service. This service runs inference on batch data and stops automatically when done. Batch inference works well for offline tasks like big data analysis, batch data labeling, and model evaluation.

This section describes how to deploy a model as a batch inference service on ModelArts and how to check the prediction results.

Billing

Deploying a service in ModelArts uses compute and storage resources, which are billed. Compute resources are billed for running the inference service. Storage

resources are billed for storing data in OBS. For details, see [Inference Deployment Billing Items](#).

Prerequisites

- A ModelArts model in the **Normal** state is available.
- Data to be processed in batches has been uploaded to OBS.
- At least one empty folder has been created in OBS for storing the output.

Context

- You can create up to 1,000 batch services.
- Based on the input request (JSON or file) defined by the model, different parameters are entered. If the model input is a JSON file, a configuration file is required to generate a mapping file. If the model input is a file, no mapping file is required.

Procedure

1. Log in to the ModelArts console. In the navigation pane, choose **Model Deployment > Batch Services**.
2. Click **Deploy** in the upper left corner.
3. Configure parameters.
 - a. Enter basic information, including **Name** and **Description**. A name is generated by default, for example, **service-bc0d**, which you can modify.
 - b. Configure other parameters, including the resource pool and model configurations.

Table 8-25 Parameters

Parameter	Description
Resource Pool	Public Resource Pool CPU and GPU public resource pools are available for you to select. To use a public resource pool, contact the administrator to create one.
	Dedicated Resource Pool Select a specification from the resource pool specifications.
Model Source	Choose My Model or My Subscriptions as needed.
Model and Version	Select the model and version that are in the Normal state.

Parameter	Description
Input Path	Select the OBS directory where the uploaded data is stored. Select a folder or a .manifest file. For details about the specifications of the .manifest file, see Manifest File Specifications. • If the input data is an image, ensure that the size of a single image is less than 12 MB. • If the input data is in CSV format, ensure that no Chinese character is included. • If the input data is in CSV format, ensure that the file size does not exceed 12 MB. • If an image or CSV file is larger than 12 MB, an error is reported. In this case, resize the file or contact technical support to adjust the file size limit.
Request Path	URL used for calling the model API in a batch service, and also the request path of the model service. Its value is obtained from the url field of apis in the model configuration file.
Mapping Relationship	If the model input is in JSON format, the system automatically generates the mapping based on the configuration file corresponding to the model. If the model input is other file, mapping is not required. The mapping file is generated automatically. Enter the field index corresponding to each parameter in the CSV file. The index starts from 0. Mapping rule: The mapping rule comes from the input parameter (request) in the model configuration file config.json. When type is set to string, number, integer, or boolean, you are required to set the index parameter. For details about the mapping rule, see Mapping Example. The index must be a positive integer starting from 0. If the value of index does not comply with the rule, this parameter is ignored in the request. After the mapping rule is configured, the CSV data must be separated by commas (,).
Output Path	The path for storing the batch prediction results. You can select an empty folder you created.
Instance Flavor	The system provides available compute resources matching your model. Select an available resource from the drop-down list.

Parameter	Description
Instances	Number of instances for the current model version. If you set the number of nodes to 1 , the standalone computing mode is used. If you set the number of nodes to a value greater than 1, the distributed computing mode is used. Select a computing mode based on your actual needs.
Environment Variable	Set environment variables and inject them to the pod. To ensure data security, do not enter sensitive information, such as plaintext passwords, in environment variables.
Timeout	Timeout of a single model, including both the deployment and startup time. The default value is 20 minutes. The value must range from 3 to 120.
Runtime Log Output	This feature is disabled by default. The runtime logs of batch services are stored only in the ModelArts log system. You can query the runtime logs in the Logs tab of the service details page. If this feature is enabled, the runtime logs of batch services will be exported and stored in Log Tank Service (LTS). LTS automatically creates log groups and log streams and caches run logs generated within seven days by default. For details about LTS log management, see Log Tank Service. • This cannot be disabled once it is enabled. • You will be billed for log query and storage features provided by LTS. For details, see LTS Pricing Details. WARNING Do not print unnecessary audio log files. Otherwise, system logs may fail to be displayed, and the error message "Failed to load audio" may be displayed.

4. Confirm the configurations and complete service deployment as prompted. Deploying a service generally requires a period of time, which may be several minutes or tens of minutes depending on the amount of your data and resources.

 NOTE

Once a batch service is deployed, it will start immediately. You will be billed for the chosen resources while it is running.

You can go to the batch service list to view the basic information about the batch service. In the batch service list, after the status of the newly deployed service changes from **Deploying** to **Running**, the service is deployed.

Manifest File Specifications

ModelArts batch services support manifest files, which describe data input and output.

Example input manifest file

- File name: **test.manifest**
- File content:

```
{"source": "obs://test/data/1.jpg"}
{"source": "s3://test/data/2.jpg"}
{"source": "https://infern-data.obs.cn-north-1.myhuaweicloud.com:443/xgboostdata/data.csv?
AccessKeyId=2Q0V0TQ461N26DDL18RB&Expires=1550611914&Signature=wZBttZj5QZrReDhz1uDzwve
8GpY%3D&x-obs-security-token=gQpzb3V0aGNoaW5hixvY8V9a1SnsxmGoHYmB1SArYMyqnQT-
ZaMSxHvl68kKLAY5feYvLDM..."}
```
- Requirements on the file:
 - a. The file name extension must be **.manifest**.
 - b. The file content must be in JSON format. Each line describes a piece of input data, which must be a specific file instead of a folder.
 - c. JSON content requires a **source** field, which must be an OBS file address in either of the following formats:
 - i. Bucket path *<obs path>{{Bucket name}}/{{Object name}}/File name*, which is used to access your OBS data. You can obtain the path by accessing the OBS object. *<obs path>* can be **obs://** or **s3://**.
 - ii. Share link generated by OBS, including signature information. It applies to accessing OBS data of other users. The link has a validity period. Perform operations within the period.

Example output manifest file

A manifest file will be generated in the output directory of the batch service.

- Assume that the output path is **//test-bucket/test/**. The result is stored in the following path:

OBS bucket or directory name

```
test-bucket
├── test-bucket
│   └── test
│       ├── infer-result-{{task_id}}.manifest
│       ├── infer-result
│       │   ├── 1.jpg_result.txt
│       │   └── 2.jpg_result.txt
```

- Content of the **infer-result-0.manifest** file:

```
{"source": "obs://obs-data-bucket/test/data/1.jpg", "result": "SUCCESSFUL", "inference-loc": "obs://test-
bucket/test/infer-result/1.jpg_result.txt"}
{"source": "s3://obs-data-bucket/test/data/2.jpg", "result": "FAILED", "error_message": "Download file
failed."}
{"source": "https://infern-data.obs.example.com:443/xgboostdata/2.jpg?
AccessKeyId=2Q0V0TQ461N26DDL18RB&Expires=1550611914&Signature=wZBttZj5QZrReDhz1uDzwve
8GpY%3D&x-obs-security-token=gQpzb3V0aGNoaW5hixvY8V9a1SnsxmGoHYmB1SArYMyqnQT-
ZaMSxHvl68kKLAY5feYvLDMNZWxzhBZ6Q-3HcoZMh9glSwQOVbwm4ZytB_m8sg1fL6isU7T3CnoL9jmv
DGgT9VBC7dC1EyfSJrUcqfB_N0ykCsfrA1Tt_IQYzFDu_HyqVl-
GunUcTVdDfWICV3TrYcpmznZjliAnYUO89kAwCYGeRZsCsC0ePu4PHMsBvYV9gWmN9AUZIDn1sfRL4vo
BpwQnp6tnAgHW49y5a6hP2hCAoQ-95SpUriJ434QlymoeKfTHVMK0eZxZea-
JxOvevOCGI5CcGehEJaz48sgH81UiH2l21zocNB_hpFus2jY6KPglEjXmV6Kwmro-
ZBXWuSJUDOnSYXI-3ciYjg9-
h10b8W3sW1mOTFCWNGoWsd74it7L_5-7UUholeyPByO_REwkur2FOJsuMpGlRaPyglZxXm_jfdLFXobYtz
Zhbul4yWXga6oxTokfcwykTOYHONPoPrt5MYGYweOXXxfs3d5w2rd0y7p0QYhyTzlk5Clz7FIWNapFISL
7zdhs8RfchTqESq94KgeqatSF_ilvYmMW2r8P8x2k_eb6NJ7U_q5ztMbO9oWEcfr0D2f7n7BL_nb2HIB_H9tj
zKqvngwaimYhBbMRPfbvttW86GiwVP8vrC27FOn39Be9z2hSfJ_8pHej0yMlyNqZ481FQ5vWT_vFV3JHM-
711ZB0_hldaHfltm-J69cTfHSEOzt7DgaMIES1o7U3w%3D%3D", "result": "SUCCESSFUL", "inference-loc":
"obs://test-bucket/test/infer-result/2.jpg_result.txt"}
```

- File format:
 - a. The file name is **infer-result-{{task_id}}.manifest**, where **task_id** is the batch task ID, which is unique for a batch service.
 - b. If a large number of files need to be processed, multiple manifest files may be generated with the same suffix **.manifest** and are distinguished by suffix, for example, **infer-result-{{task_id}}_1.manifest**.
 - c. The **infer-result-{{task_id}}** directory is created in the manifest directory to store the file processing result.
 - d. The file content is in JSON format. Each line describes the output result of a piece of input data.
 - e. The JSON file contains multiple fields:
 - i. **source**: input data description, which is the same as that of the input manifest file
 - ii. **result**: file processing result, which can be **SUCCESSFUL** or **FAILED**
 - iii. **inference-loc**: output result path. This field is available when result is **SUCCESSFUL**. The format is **obs://{{Bucket name}}/{{Object name}}**.
 - iv. **error_message**: error information. This field is available when the result is **FAILED**.

Mapping Example

The following example shows the relationship between the configuration file, mapping rule, CSV data, and inference request.

The following uses a file for prediction as an example:

```
[
  {
    "method": "post",
    "url": "/",
    "request": {
      "Content-type": "multipart/form-data",
      "data": {
        "type": "object",
        "properties": {
          "data": {
            "type": "object",
            "properties": {
              "req_data": {
                "type": "array",
                "items": [
                  {
                    "type": "object",
                    "properties": {
                      "input_1": {
                        "type": "number"
                      },
                      "input_2": {
                        "type": "number"
                      },
                      "input_3": {
                        "type": "number"
                      },
                      "input_4": {
                        "type": "number"
                      }
                    }
                  }
                ]
              }
            }
          }
        }
      }
    }
  }
]
```

```
}  
}  
}  
}  
}  
}  
}
```

The ModelArts console automatically resolves the mapping relationship from the configuration file as shown below. When calling a ModelArts API, configure the mapping by following the rule.

```
{  
  "type": "object",  
  "properties": {  
    "data": {  
      "type": "object",  
      "properties": {  
        "req_data": {  
          "type": "array",  
          "items": [  
            {  
              "type": "object",  
              "properties": {  
                "input_1": {  
                  "type": "number",  
                  "index": 0  
                },  
                "input_2": {  
                  "type": "number",  
                  "index": 1  
                },  
                "input_3": {  
                  "type": "number",  
                  "index": 2  
                },  
                "input_4": {  
                  "type": "number",  
                  "index": 3  
                }  
              }  
            }  
          ]  
        }  
      }  
    }  
  }  
}
```

The following shows the format of the CSV data for inference. The data must be separated by commas (,).

```
5.1,3.5,1.4,0.2  
4.9,3.0,1.4,0.2  
4.7,3.2,1.3,0.2
```

Depending on the defined mapping relationship, the inference request is shown below, whose format is similar to that for real-time services.

```
{  
  "data": {  
    "req_data": [{  
      "input_1": 5.1,  
      "input_2": 3.5,  
      "input_3": 1.4,  
      "input_4": 0.2  
    }  
  ]  
}
```

```
}  
}
```

Viewing the Batch Service Prediction Result

When deploying a batch service, you can select the location of the output data directory. You can view the running result of the batch service that is in the **Completed** state.

- Step 1** Log in to the ModelArts console and choose **Model Deployment > Batch Services**.
- Step 2** Click the name of the target service in the **Completed** status. The service details page is displayed.
- You can view the service name, status, ID, input path, output path, and description.
 - You can click  next to **Description** to edit the description.
- Step 3** Obtain the detailed OBS path next to **Output Path**, switch to the path and obtain the batch service prediction results, including the prediction result file and the model prediction result.

If the prediction is successful, the directory contains the prediction result file and model prediction result. Otherwise, the directory contains only the prediction result file.

- Prediction result file: The file is in the *xxx.manifest* format and contains the file path and prediction result.
- Model prediction output:
 - If images are input, a result file is generated for each image in the *Image name_result.txt* format, for example, **IMG_20180919_115016.jpg_result.txt**.
 - If audio files are input, a result file is generated for each audio file in the *Audio file name_result.txt* format, for example, **1-36929-A-47.wav_result.txt**.
 - If table data is input, the result file is generated in the *Table name_result.txt* format, for example, **train.csv_result.txt**.

----End

8.6 Managing ModelArts Models

8.6.1 Viewing ModelArts Model Details

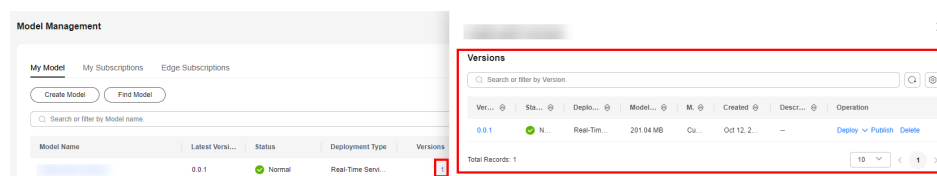
Viewing the Model List

You can view all created models on the model list page. The model list page displays the following information.

Table 8-26 Model list

Parameter	Description
Model Name	Model name.
Latest Version	Latest version of a model.
Status	Model status.
Deployment Type	Types of the services that a model can be deployed as.
Versions	Number of model versions.
Request Mode	Request mode of real-time services. • Synchronous request: one-off inference with results returned synchronously (within 60s). This mode is suitable for images and small video files. • Asynchronous request: one-off inference with results returned asynchronously (longer than 60s). This mode is suitable for real-time video inference and large videos.
Created	Model creation time.
Description	Model description.
Operation	• Deploy: Deploy a model as real-time services, edge services, or batch services. • Create Version: Create a model version. The settings of the last version are used by default, except for the version. You can change the parameter settings. • Delete: Delete a model. NOTE If a model version has been deployed as a service, you must delete the associated service before deleting the model version. A deleted model cannot be recovered.

Click the number in **Versions** to view the version list.

Figure 8-45 Version list

The version list displays the following information.

Table 8-27 Version list

Parameter	Description
Version	Current version of a model.
Status	Model status.
Deployment Type	Types of the services that a model can be deployed as.
Model Size	Model size.
Model Source	Model source.
Created	Model creation time.
Description	Model description.
Operation	• Deploy: Deploy a model as real-time services, edge services, or batch services. • Delete: Delete a version of a model.

Viewing Model Details

After a model is created, you can view the model information on the model details page.

1. Log in to the ModelArts console, and choose **Model Management** from the navigation pane.
2. Click the name of the target model to access its details page.

On the model details page, you can view the basic information and precision of the model, and switch tab pages to view more information.

Table 8-28 Basic model information

Parameter	Description
Name	Model name.
Status	Model status.
Version	Current version of a model.
ID	Model ID.
Description	Click the edit button to add the description of a model.
Deployment Type	Types of the services that a model can be deployed as.
Meta Model Source	Source of the meta model, which can be training jobs, OBS, or container images.
Training Name	Associated training job if the meta model comes from a training job. Click the training job name to go to its details page.

Parameter	Description
Training Version	Training job version if the meta model comes from an old-version training job.
Storage path of the meta model	Path to the meta model if the meta model comes from OBS.
Container Image Storage Path	Path to the container image if the meta model comes from a container image.
AI Engine	AI engine if the meta model comes from a training job or OBS.
Engine Package Address	Engine package address if the meta model comes from OBS and AI Engine is Custom .
Runtime Environment	Runtime environment on which the meta model depends if the meta model comes from a training job or OBS and a preset AI engine is used.
Container API	Protocol and port number for starting the model if the meta model comes from OBS (AI Engine is Custom) or a container image.
Inference Code	Path to the inference code if the meta model comes from an old-version training job.
Image Replication	Image replication status for meta models from a container image.
Dynamic loading	Dynamic loading status if the meta model comes from a training job or OBS.
Size	Model size.

Parameter	Description
Health Check	Displays health check status if the meta model comes from OBS or a container image. When health check is enabled, the probe parameter settings are displayed. • Startup Probe: This probe checks if the application instance has started. If a startup probe is provided, all other probes are disabled until it succeeds. If the startup probe fails, the instance is restarted. If no startup probe is provided, the default status is Success. • Readiness Probe: This probe verifies whether the application instance is ready to handle traffic. If the readiness probe fails (meaning the instance is not ready), the instance is taken out of the service load balancing pool. Traffic will not be routed to the instance until the probe succeeds. • Liveness Probe: This probe monitors the application health status. If the liveness probe fails (indicating the application is unhealthy), the instance is automatically restarted. The probe parameters include Check Mode, Health Check URL (displayed when Check Mode is set to HTTP request), Health Check Command (displayed when Check Mode is set to Command), Health Check Period, Delay, Timeout, and Maximum Failures.
Model Description	Description document added during the creation of a model.
Instruction Set Architecture	System architecture.
Inference Accelerator	Type of inference accelerator cards.

Table 8-29 Model details tabs

Parameter	Description
Model Precision	Model recall, precision, accuracy, and F1 score of a model.
Parameter Configuration	API configuration, input parameters, and output parameters of a model.
Runtime Dependency	Model dependency on the environment. If creating a job failed, edit the runtime dependency. After the modification is saved, the system will automatically use the original image to create the job again.

Parameter	Description
Events	The progress of key operations during model creation. Events are stored for three months and will be automatically cleared then. For details about how to view events of a model, see Viewing ModelArts Model Events .
Constraint	Displays the constraints of service deployment, such as the request mode, boot command, and model encryption, based on the settings during model creation. For models in asynchronous request mode, parameters including the input mode, output mode, service startup parameters, and job configuration parameters can be displayed.
Associated Services	The list of services that a model was deployed. Click a service name to go to the service details page.

8.6.2 Viewing ModelArts Model Events

During the creation of a model, every key event is automatically recorded. You can view the events on the details page of the model at any time.

The following table lists the events, based on which you can locate faults occurred during model creation. The following table lists the available events.

Type	Event (xxx should be replaced with the actual value.)	Solution
Normal	The model starts to import.	N/A
Abnormal	Failed to create the image.	Locate and rectify the fault based on the error information. FAQs
Abnormal	The custom image does not support specified dependencies.	The runtime dependencies cannot be configured when a custom image is imported. Install the pip dependency package in the Dockerfile that is used to create the image. FAQs

Type	Event (<i>xxx</i> should be replaced with the actual value.)	Solution
Abnormal	Only custom images support swr_location .	Delete the swr_location field from the model configuration file config.json and try again.
Abnormal	The health check API of a custom image must be <i>xxx</i> .	Modify the health check API of the custom image and try again.
Normal	The image creation task is in the <i>xxx</i> state.	N/A
Abnormal	Label <i>xxx</i> does not exist in image <i>xxx</i> .	Contact technical support.
Abnormal	Invalid parameter value <i>xxx</i> exists in the model configuration file.	Delete invalid parameters from the model configuration file and try again.
Abnormal	Failed to obtain the labels of image <i>xxx</i> .	Contact technical support.
Abnormal	Failed to import data because <i>xxx</i> is larger than <i>xxx</i> GB.	The size of the model or image exceeds the upper limit. Downsize the model or image and import it again. FAQs
Abnormal	User <i>xxx</i> does not have OBS permission <code>obs:object:PutObjectAcl</code> .	The IAM user does not have the <code>obs:object:PutObjectAcl</code> permission on OBS. Add the agency permission for the IAM user. FAQs
Abnormal	Creating the image timed out. The timeout duration is <i>xxx</i> minutes.	There is a timeout limit for image building using ImagePacker. Simplify the code to improve efficiency. FAQs
Normal	Model description updated.	N/A

Type	Event (xxx should be replaced with the actual value.)	Solution
Normal	Model runtime dependencies not updated.	N/A
Normal	Model runtime dependencies updated. Recreating the image.	N/A
Abnormal	SWR traffic control triggered. Try again later.	SWR traffic control triggered. Try again later.
Normal	The system is being upgraded. Try again later.	N/A
Abnormal	Failed to obtain the source image. An error occurred in authentication. The token has expired.	Contact technical support.
Abnormal	Failed to obtain the source image. Check whether the image exists.	Contact technical support.
Normal	Source image size calculated.	N/A
Normal	Source image shared.	N/A
Abnormal	Failed to create the image due to traffic control. Try again later.	Traffic control triggered. Try again later.
Abnormal	Failed to send the image creation request.	Contact technical support.
Abnormal	Failed to share the source image. Check whether the image exists or whether you have the permission to share the image.	Check whether the image exists or whether you have the permission to share the image.
Normal	The model imported.	N/A
Normal	Model file imported.	N/A
Normal	Model size calculated.	N/A
Abnormal	Failed to import the model.	For details about how to locate and rectify the fault, see FAQs .
Abnormal	Failed to copy the model file. Check whether you have the OBS permission.	Check whether you have the OBS permission. FAQs
Abnormal	Failed to schedule the image creation task.	Contact technical support.

Type	Event (xxx should be replaced with the actual value.)	Solution
Abnormal	Failed to start the image creation task.	Contact technical support.
Abnormal	The Roman image has been created but cannot be shared with resource tenants.	Contact technical support.
Normal	Image created.	N/A
Normal	The image creation task started.	N/A
Normal	The environment image creation task started.	N/A
Normal	The request for creating an environment image received.	N/A
Normal	The request for creating an image received.	N/A
Normal	An existing environment image is used.	N/A
Abnormal	Failed to create the image. For details, see image creation logs.	View the build logs to locate and rectify the fault. FAQs
Abnormal	Failed to create the image due to an internal system error. Contact technical support.	Contact technical support.
Abnormal	Failed to import model file xxx because it is larger than 5 GB.	The size of the model file xxx is greater than 5 GB. Downsize the model file and try again, or use dynamic loading to import the model file. FAQs
Abnormal	Failed to create the OBS bucket due to an internal system error. Contact technical support.	Contact technical support.
Abnormal	Failed to calculate the model size. Subpath xxx does not exist in path xxx.	Correct the subpath and try again, or contact technical support.

Type	Event (xxx should be replaced with the actual value.)	Solution
Abnormal	Failed to calculate the model size. The model of the xxx type does not exist in path xxx.	Check the storage location of the model of the xxx type, correct the path, and try again, or contact technical support.
Warning	Failed to calculate the model size. More than one xxx model file is stored in path xxx.	N/A

During model creation, key events can both be manually and automatically refreshed.

Viewing Events

1. Log in to the ModelArts console. In the navigation pane on the left, choose **Model Management**. In the model list, click the name of the target model to go to its details page.
2. View the events in the **Events** tab.

8.6.3 Managing ModelArts Model Versions

For model lineage and tuning, ModelArts provides model versioning.

Prerequisites

You have created a model in ModelArts.

Creating a Version

On the **Model Management** page, click **Create Version** in the **Operation** column of the target model. On the **Create Version** page, configure parameters. For details, see [Creating a Model](#). Click **Create now**.

Deleting a Version

On the **Model Management** page, click the number in **Versions**. Click **Delete** in the **Operation** column of the target version and enter **DELETE**.

NOTE

If a model version has been deployed as a service, you must delete the associated service before deleting the model version. A deleted version cannot be recovered.

Deleting a Model

On the **Model Management** page, click **Delete** in the **Operation** column of the target model, enter **DELETE** in the text box, and click **OK** to delete the model.

NOTE

If a model version has been deployed as a service, you must delete the associated service before deleting the model version. A deleted model cannot be recovered.

8.7 Managing a Synchronous Real-Time Service

8.7.1 Viewing Details About a Real-Time Service

After a model is deployed as a real-time service, you can access the service page to view its details.

1. Log in to the ModelArts console and choose **Model Deployment > Real-Time Services**.
2. Click the name of the target service to access its details page.
View service information. For details, see [Table 8-30](#).

Table 8-30 real-time service parameters

Parameter	Description
Name	Name of the real-time service.
Status	Status of the real-time service.
Source	AI application source of the real-time service.
Service ID	Real-time service ID.
Description	Service description, which you can click the edit button to modify.
Resource Pool	Resource pool specifications used by the service. If the public resource pool is used for deployment, this parameter is not displayed.
Custom Settings	Customized configurations based on real-time service versions. This allows version-based traffic distribution policies and configurations. Enable this option and click View Settings to customize the settings. For details, see Modifying Customized Settings .
Traffic Limit	Maximum number of times a service can be accessed within a second.

Parameter	Description
Runtime Log Output	This feature is disabled by default. The runtime logs of real-time services are stored only in the ModelArts log system. If this feature is enabled, the runtime logs of real-time services will be exported and stored in Log Tank Service (LTS). LTS automatically creates log groups and log streams and caches run logs generated within seven days by default. For details about LTS log management, see Log Tank Service. NOTE • This cannot be disabled once it is enabled. • You will be billed for log query and storage features provided by LTS. For details, see LTS Pricing Details. • Do not print unnecessary audio log files. Otherwise, system logs may fail to be displayed, and the error message "Failed to load audio" may be displayed.
WebSocket	Whether to upgrade to the WebSocket service.

3. Switch between tabs on the details page of a real-time service to view more details. For details, see [Table 8-31](#).

Table 8-31 Details of a real-time service

Parameter	Description
Usage Guides	This page displays the API URL, model information, input parameters, and output parameters. You can click  to copy the API URL to call the service. If application authentication is supported, you can view the API URL and authorization management details, including the application name, AppKey, and AppSecret, in the Usage Guides. You can also add or cancel authorization for an application.
Prediction	You can perform real-time prediction on this page. For details, see Testing Real-Time Service Prediction.
Instance	Instance information of the asynchronous real-time service. The number of instances is the same as the number of instances set during service deployment. If the service is modified or the service is abnormal, the number of instances changes. To rebuild an abnormal instance, click Delete. After the instance is deleted, a new instance with the same compute specifications is automatically created.

Parameter	Description
Configuration Updates	This page displays Current Configurations and Update History. • Current Configurations: model name, version, status, instance flavor, traffic ratio, number of instances, deployment timeout interval, environment variables, and storage mounting. If the service is deployed in a dedicated resource pool, the resource pool information is also displayed. • Update History: historical model information.
Monitoring	This page displays resource usage and model calls. • Resource Usage: includes the used and available CPU, memory, GPU, and NPU resources. • AI Application Calls: indicates the number of model calls. The statistics collection starts after the model status changes to Ready. (This parameter is not displayed for WebSocket services.)
Events	This page displays key operations during service use, such as the service deployment progress, detailed causes of deployment exceptions, and time points when a service is started, stopped, or modified. Events are saved for one month and will be automatically cleared then. For details about how to view events of a service, see Viewing Events of a Real-Time Service.

Parameter	Description
Logs	This page displays the log information about each model in the service. You can view logs generated in the latest 5 minutes, latest 30 minutes, latest 1 hour, and user-defined time segment. You can select the start time and end time when defining a time segment. If this feature is enabled, the logs stored in LTS will be displayed. You can click View Complete Logs on LTS to view all logs. Log search rules: • Do not enter a string that contains any of the following delimiters: ,";=()[]{}@<>/:\\n\\t\\r. • You can use exact search by keyword. A keyword refers to the word between two adjacent delimiters. • You can use fuzzy search by keyword. For example, you can enter error, er?or, rro*, or er*r. • You can enter a phrase for exact search. For example, Start to refresh. • Before enabling this feature, you can combine keywords with && or . For example, query logs&&erro* or query logs erro*. After enabling this feature, you can combine keywords with AND or OR. For example, query logs AND erro* or query logs OR erro*.
Tags	Tags that have been added to the service. Tags can be added, modified, and deleted. For details about how to use tags, see Using TMS Tags to Manage Resources by Group
Cloud Shell	You can use Cloud Shell provided by the ModelArts console to log in to the instance container of a running real-time service. For details, see Using Cloud Shell to Debug a Real-Time Service Instance Container.

Modifying Customized Settings

A customized configuration rule consists of the configuration condition (**Setting**), access version (**Version**), and customized running parameters (including **Setting Name** and **Setting Value**).

You can configure different settings with customized running parameters for different versions of a real-time service.

The priorities of customized configuration rules are in descending order. You can change the priorities by dragging the sequence of customized configuration rules.

After a rule is matched, the system will no longer match subsequent rules. A maximum of 10 configuration rules can be configured.

Table 8-32 Parameters for Custom Settings

Parameter	Mandatory	Description
Setting	Yes	Expression of the Spring Expression Language (SpEL) rule. Only the equal, matches, and hashCode expressions of the character type are supported.
Version	Yes	Access version for a customized service configuration rule. When a rule is matched, the real-time service of the version is requested.
Setting Name	No	Key of a customized running parameter, consisting of a maximum of 128 characters. Configure this parameter if the HTTP message header is used to carry customized running parameters to a real-time service.
Setting Value	No	Value of a customized running parameter, consisting of a maximum of 256 characters. Configure this parameter if the HTTP message header is used to carry customized running parameters to a real-time service.

Customized settings can be used in the following scenarios:

- If multiple versions of a real-time service are deployed for gray release, customized settings can be used to distribute traffic by user.

Table 8-33 Built-in variables

Built-in Variable	Description
DOMAIN_NAME	Account name used to call an inference request
DOMAIN_ID	Account ID used to call an inference request
PROJECT_NAME	Project name used to call an inference request
PROJECT_ID	Project ID used to call an inference request
USER_NAME	Username used to call an inference request
USER_ID	User ID used to call an inference request

Pound key (#) indicates that a variable is referenced. The matched character string must be enclosed in single quotation marks.

```
#{Built-in variable} == 'Character string'  
#{Built-in variable} matches 'Regular expression'
```

- Example 1:
If the account name in the inference request is **User A**, the specified version is matched.
`#DOMAIN_NAME == 'zhangsan'`
- Example 2:
If the account name in the inference request starts with **op**, the specified version is matched.
`#DOMAIN_NAME matches 'op.*'`

Table 8-34 Common regular expressions

Character	Description
.	Match any single character except <code>\n</code> . To match any character including <code>\n</code> , use <code>(. \n)</code> .
*	Match the subexpression that it follows for zero or multiple times. For example, <code>zo*</code> can match <code>z</code> and <code>zoo</code> .
+	Match the subexpression that it follows for once or multiple times. For example, <code>zo+</code> can match <code>zo</code> and <code>zoo</code> , but cannot match <code>z</code> .
?	Match the subexpression that it follows for zero or one time. For example, <code>do(es)?</code> can match <code>does</code> or <code>do</code> in <code>does</code> .
^	Match the start of the input string.
\$	Match the end of the input string.
{n}	<i>n</i> is a non-negative integer, which matches exactly <i>n</i> number of occurrences of an expression. For example, <code>o{2}</code> cannot match <code>o</code> in <code>Bob</code> , but can match two <code>os</code> in <code>food</code> .
x y	Match x or y. For example, <code>z food</code> can match <code>z</code> or <code>food</code> , and <code>(z f)ood</code> can match <code>zood</code> or <code>food</code> .
[xyz]	Character set, where any single character in it can be matched. For example, <code>[abc]</code> can match <code>a</code> in <code>plain</code> .

Figure 8-46 Traffic distribution by user

Custom Settings ×

Setting ?	Version ?	Setting Name (Optional) ?	Setting Value (Optional) ?	
#HEADER_version == '0.0.1'	0.0.2			
#HEADER_testheader matches 'mocl'	0.0.1			

⊕ Add

Cancel OK

- If multiple versions of a real-time service are deployed for gated launch, customized settings can be used to access different versions through the header.

Start with **#HEADER_**, indicating that the header is referenced as a condition.

```
#HEADER_{key} == '{value}'
```

```
#HEADER_{key} matches '{value}'
```

- Example 1:

If the header of an inference HTTP request contains a version and the value is **0.0.1**, the condition is met. Otherwise, the condition is not met.

```
#HEADER_version == '0.0.1'
```

- Example 2:

If the header of an inference HTTP request contains **testheader** and the value starts with **mock**, the rule is matched.

```
#HEADER_testheader matches 'mock.*'
```

- Example 3:

If the header of an inference HTTP request contains **uid** and the hash code value meets the conditions described in the following algorithm, the rule is matched.

```
#HEADER_uid.hashCode() % 100 < 10
```

Figure 8-47 Using the header to access different versions

Setting	Version	Setting Name (Optional)	Setting Value (Optional)
#HEADER_version == '0.0.1'	0.0.2		
#HEADER_testheader matches 'mock'	0.0.1		

[Add](#)

Cancel OK

- If a real-time service version supports different runtime configurations, you can use **Setting Name** and **Setting Value** to specify customized runtime parameters so that different users can use different running configurations.

Example:

When user A accesses the model, the user uses configuration A. When user B accesses the model, the user uses configuration B. When matching a running configuration, ModelArts adds a header to the request and also the customized running parameters specified by **Setting Name** and **Setting Value**.

Figure 8-48 Customized running parameters added for a customized configuration rule

Custom Settings ×

Setting ?	Version ?	Setting Name (Optional) ?	Setting Value (Optional) ?	
#HEADER_NAME == 'zhangsan'	0.0.2 ▼	testkey1	testvalue1	🗑
#HEADER_NAME == 'lisi'	0.0.2 ▼	testkey2	testvalue2	🗑

⊕ Add

Cancel OK

8.7.2 Viewing Events of a Real-Time Service

During the whole lifecycle of a service, every key event is automatically recorded. You can view the events on the details page of the service at any time.

This helps you better understand the service deployment and running process and accurately locate faults when a task exception occurs. You can check the following events:

Table 8-35 Events

Event Type	Event (xxx should be replaced with the actual value.)	Solution
Normal	The service starts to deploy.	N/A
Abnormal	Insufficient resources. Wait until idle resources are sufficient.	Wait until the resources are released and try again.
Abnormal	Insufficient xxx. The scheduling failed. Supplementary information: xxx	Learn about resource insufficiency details based on the supplementary information. For details, see FAQs .
Normal	The image starts to create.	N/A
Abnormal	Failed to create model image xxx. For details, see logs : nxxx.	Locate and rectify the fault based on the build logs.
Abnormal	Failed to create the image.	Contact technical support.

Event Type	Event (xxx should be replaced with the actual value.)	Solution
Normal	The image is created.	N/A
Abnormal	Service xxx failed. Error: xxx	Locate and rectify the fault based on the error information.
Abnormal	Failed to update the service. Perform a rollback.	Contact technical support.
Normal	The service is being updated.	N/A
Normal	The service is being started.	N/A
Normal	The service is being stopped.	N/A
Normal	The service has been stopped.	N/A
Normal	Auto stop has been disabled.	N/A
Normal	Auto stop has been enabled. The service will stop after xs.	N/A
Normal	The service stops when the auto stop time expires.	N/A
Abnormal	The service is stopped because the quota exceeds the upper limit.	Contact technical support.
Abnormal	Failed to automatically stop the service. Error: xxx	Locate and rectify the fault based on the error information.
Normal	Service instances deleted from resource pool xxx.	N/A
Normal	Service instances stopped in resource pool xxx.	N/A
Abnormal	The batch service failed. Try again later. Error: xxx	Locate and rectify the fault based on the error information.
Normal	The service has been executed.	N/A

Event Type	Event (xxx should be replaced with the actual value.)	Solution
Abnormal	Failed to stop the service. Error: xxx	Locate and rectify the fault based on the error information.
Normal	The subscription license xxx is to expire.	N/A
Normal	Service xxx started.	N/A
Abnormal	Failed to start service xxx.	For details about how to locate and rectify the fault, see FAQs .
Abnormal	Service deployment timed out. Error: xxx	Locate and rectify the fault based on the error information.
Normal	Failed to update the service. The update has been rolled back.	N/A
Abnormal	Failed to update the service. The rollback failed.	Contact technical support.
Normal	[model 0.0.1] OBS bucket, OBS parallel file system, or SFS Turbo mounted successfully. [%s] %s volume successfully.	N/A

During service deployment and running, key events can both be manually and automatically refreshed.

Viewing Events

1. In the navigation pane of the ModelArts console, choose **Model Deployment** > **Real-Time Services**. In the service list, click the name or ID of the target service to go to its details page.
2. View the events in the **Events** tab.

8.7.3 Managing the Lifecycle of a Real-Time Service

Starting a Service

A service not in the **Deploying** state can be started. A service is billed from the time when it is running. To start a service, use either of the following methods:

- Log in to the ModelArts console and choose **Model Deployment** from the navigation pane. Go to the service management page of the target service. Click **Start** in the **Operation** column to start the target service.
- Log in to the ModelArts console and choose **Model Deployment** from the navigation pane. Go to the service management page of the target service. Click the name of the target service to access its details page. Click **Start** in the upper right corner of the page to start the service.

 NOTE

Services deployed on ModelArts edge nodes or ModelArts edge resource pools cannot be started.

Stopping a Service

A stopped service will no longer be billed. Stop a service in either of the following ways:

- Log in to the ModelArts console and choose **Model Deployment > Real-Time Services**. Choose **More > Stop** in the **Operation** column to stop the target service.
- Log in to the ModelArts console and choose **Model Deployment > Real-Time Services**. Click the name of the target service to access its details page. Click **Stop** in the upper right corner of the page to stop the service.

 NOTE

Services deployed on ModelArts edge nodes or ModelArts edge resource pools cannot be stopped.

Deleting a Service

If a service is no longer in use, delete it to release resources.

Log in to the ModelArts console and choose **Model Deployment > Real-Time Services**.

- In the real-time service list, choose **More > Delete** in the **Operation** column of the target service to delete it.
- Select services in the real-time service list and click **Delete** above the list to delete services in batches.
- Click the name of the target service. On the displayed service details page, click **Delete** in the upper right corner to delete the service.

 NOTE

- A deleted service cannot be recovered.
- A service cannot be deleted without agency authorization.
- If **Advanced Log Management** is enabled for a real-time service, you are advised to delete the LTS logs and streams when you delete the service. This prevents additional fees incurred by the logs and streams.

Restarting a Service

You can restart a real-time service only when the service is in the **Running** or **Alarm** state. Batch services and edge services cannot be restarted. You can restart a real-time service in either of the following ways:

- Log in to the ModelArts console and choose **Model Deployment** > **Real-Time Services**. Choose **More** > **Restart** in the **Operation** column to restart the target service.
- Log in to the ModelArts console and choose **Model Deployment** > **Real-Time Services**. Click the name of the target service to access its details page. Click **Restart** in the upper right corner of the page to restart the service.

NOTE

Services deployed on ModelArts edge nodes or ModelArts edge resource pools cannot be restarted.

8.7.4 Modifying a Real-Time Service

For a deployed service, you can modify its basic information to match service changes and change the model version to upgrade it.

You can modify the basic information about a service in either of the following ways:

Method 1: Modify the Service Information on the Service Management Page

Method 2: Modify the Service Information on the Service Details Page

Prerequisites

The service has been deployed. The service in the **Deploying** state cannot be upgraded by modifying the service information.

Constraints

- Improper upgrade operations will interrupt services during the upgrade.
- ModelArts supports hitless rolling upgrade of real-time services in some scenarios. Prepare for and fully verify the upgrade.

Table 8-36 Scenarios for hitless rolling upgrade

Meta Model Source	Using a Public Resource Pool	Using a Dedicated Resource Pool
Training job	Not supported	Not supported
Container image	Not supported	Supported. The custom image for creating a model must meet Specifications for Custom Images .
OBS	Not supported	Not supported

Method 1: Modify the Service Information on the Service Management Page

1. Log in to the ModelArts console and choose **Model Deployment** from the navigation pane. Go to the service management page of the target service.
2. In the service list, click **Modify** in the **Operation** column of the target service, modify basic service information, and submit the modification task as prompted.

When some parameters are modified, the system automatically restarts the service for the modification to take effect. When you submit a service modification task, if a restart is required, a dialog box will be displayed.

For details about the real-time service parameters, see [Deploying a Model as a Real-Time Service](#). To modify a real-time service, you also need to set **Max. Invalid Instances** to set the maximum number of nodes that can be concurrently upgraded, during which time these nodes are invalid.

When modifying a real-time service's parameters, you can add a custom environment variable to migrate the service to a new resource pool.

Method 2: Modify the Service Information on the Service Details Page

1. Log in to the ModelArts console and choose **Model Deployment** from the navigation pane. Go to the service management page of the target service.
2. Click the name of the target service to access its details page.
3. Click **Modify** in the upper right corner of the page, modify the service details, and submit the modification task as prompted.

When some parameters are modified, the system automatically restarts the service for the modification to take effect. When you submit a service modification task, if a restart is required, a dialog box will be displayed.

For details about the real-time service parameters, see [Deploying a Model as a Real-Time Service](#). To modify a real-time service, you also need to set **Max. Invalid Instances** to set the maximum number of nodes that can be concurrently upgraded, during which time these nodes are invalid.

When modifying a real-time service's parameters, you can add a custom environment variable to migrate the service to a new resource pool.

8.7.5 Viewing Performance Metrics of a Real-Time Service on Cloud Eye

ModelArts Metrics

The cloud service platform provides Cloud Eye to help you better understand the statuses of ModelArts real-time services and model loads. You can use Cloud Eye to automatically monitor your ModelArts real-time services and model loads in real time and manage alarms and notifications so that you can obtain the performance metrics of ModelArts and models.

Table 8-37 ModelArts metrics

ID	Name	Description	Value Range	Unit	Number System	Monitored Object	Monitoring Interval
cpu_usage	CPU Usage	CPU usage of ModelArts Unit: %	≥ 0%	%	N/A	ModelArts model loads	1 minute
mem_usage	Memory Usage	Memory usage of ModelArts Unit: %	≥ 0%	%	N/A	ModelArts model loads	1 minute
gpu_util	GPU Usage	GPU usage of ModelArts Unit: %	≥ 0%	%	N/A	ModelArts model loads	1 minute
gpu_memory_usage	GPU Memory Usage	GPU memory usage of ModelArts Unit: %	≥ 0%	%	N/A	ModelArts model loads	1 minute
npu_util	NPU Usage	NPU usage of ModelArts Unit: %	≥ 0%	%	N/A	ModelArts model loads	1 minute
npu_memory_usage	NPU Memory Usage	NPU memory usage of ModelArts Unit: %	≥ 0%	%	N/A	ModelArts model loads	1 minute
successfully_called_times	Number of Successful Calls	Times that ModelArts services have been successfully called Unit: counts/minute	≥ counts/minute	times/minute	N/A	ModelArts model loads ModelArts real-time services	1 minute

ID	Name	Description	Value Range	Unit	Number System	Monitored Object	Monitoring Interval
failed_called_times	Number of Failed Calls	Times that ModelArts services failed to be called Unit: counts/minute	$\geq$ counts/minute	times/minute	N/A	ModelArts model loads ModelArts real-time services	1 minute
total_called_times	Total Calls	Times that ModelArts services are called Unit: counts/minute	$\geq$ counts/minute	times/minute	N/A	ModelArts model loads ModelArts real-time services	1 minute
disk_read_rate	Disk Read Rate	Disk read rate of ModelArts Unit: bits/minute	$\geq$ bits/minute	bits/min	N/A	ModelArts model loads	1 minute
disk_write_rate	Disk Write Rate	Disk write rate of ModelArts Unit: bits/minute	$\geq$ bits/minute	bits/min	N/A	ModelArts model loads	1 minute
send_bytes_rate	Uplink rate	Outbound network traffic rate of ModelArts. Unit: bits/minute	$\geq$ bits/minute	bits/min	N/A	ModelArts model loads	1 minute
recv_bytes_rate	Downlink rate	Inbound network traffic rate of ModelArts.	$\geq$ bits/minute	bits/min	N/A	ModelArts model loads	1 minute

ID	Name	Description	Value Range	Unit	Number System	Monitored Object	Monitoring Interval
req_count_2xx	2xx Responses	Number of times that the API returns a 2xx response	≥ counts/minute	times/minute	N/A	ModelArts real-time services	1 minute
req_count_4xx	4xx Errors	Number of times that the API returns a 4xx error	≥ counts/minute	times/minute	N/A	ModelArts real-time services	1 minute
req_count_5xx	5xx Errors	Number of times that the API returns a 5xx error	≥ counts/minute	times/minute	N/A	ModelArts real-time services	1 minute
avg_latency	Average Latency	Average latency of the API	≥ ms	ms	N/A	ModelArts real-time services	1 minute
tp_99	TP99	Collects the response durations of each call over the last minute, arranges them in ascending order, and then excludes the top 1% of values. The highest remaining value represents the TP99.	≥ ms	ms	N/A	ModelArts real-time services	1 minute

ID	Name	Description	Value Range	Unit	Number System	Monitored Object	Monitoring Interval
tp_999	TP99.9	Collects the response durations of each call over the last minute, arranges them in ascending order, and then excludes the top 0.1% of values. The highest remaining value represents the TP99.9.	≥ ms	ms	N/A	Model Arts real-time services	1 minute
If a monitored object has multiple dimensions, all dimensions are mandatory when you use APIs to query the metrics. - The following provides an example of using the multi-dimensional dim to query a single monitoring metric: dim.0=service_id,530cd6b0-86d7-4818-837f-935f6a27414d&dim.1="model_id,3773b058-5b4f-4366-9035-9bbd9964714a" - The following provides an example of using the multi-dimensional dim to query monitoring metrics in batches: "dimensions": [{ "name": "service_id", "value": "530cd6b0-86d7-4818-837f-935f6a27414d" } { "name": "model_id", "value": "3773b058-5b4f-4366-9035-9bbd9964714a" }]							

Table 8-38 Dimension description

Key	Value
service_id	Real-time service ID
model_id	Model ID

Setting Alarm Rules

Setting alarm rules allows you to customize the monitored objects and notification policies so that you can know the status of ModelArts real-time services and models in a timely manner.

An alarm rule includes the alarm rule name, monitored object, metric, threshold, monitoring interval, and whether to send a notification. This section describes how to set alarm rules for ModelArts services and models.

NOTE

Only real-time services in the **Running** status can be interconnected with CES.

Prerequisites:

- A ModelArts real-time service has been created.
- ModelArts monitoring has been enabled on Cloud Eye. To do so, log in to the Cloud Eye console. On the Cloud Eye page, click **Custom Monitoring**. Then, enable ModelArts monitoring as prompted.

Set an alarm rule in any of the following ways:

- Set an alarm rule for all ModelArts services.
- Set an alarm rule for a ModelArts service.
- Set an alarm rule for a model version.
- Set an alarm rule for a metric of a service or model version.

Method 1: Setting an Alarm Rule for All ModelArts Services

Step 1 Log in to the management console.

Step 2 In the **Service List**, click **Cloud Eye** under **Management & Governance**.

Step 3 In the navigation pane on the left, choose **Alarm Management > Alarm Rules** and click **Create Alarm Rule**.

Step 4 On the **Create Alarm Rule** page, set **Resource Type** to **ModelArts**, **Dimension** to **Service**, and **Method** to **Configure manually**, and set alarm policies. Then, confirm settings and click **Create**.

----End

Method 2: Setting an Alarm Rule for a Single Service

Step 1 Log in to the management console.

- Step 2** In the **Service List**, click **Cloud Eye** under **Management & Governance**.
- Step 3** In the navigation pane, choose **Cloud Service Monitoring > ModelArts**.
- Step 4** Locate a real-time service for which you want to create an alarm rule and click **Create Alarm Rule** in the **Operation** column.
- Step 5** On the **Create Alarm Rule** page, create an alarm rule for ModelArts real-time services and models as prompted.

----End

Method 3: Setting an Alarm Rule for a Model Version

- Step 1** Log in to the management console.
- Step 2** In the **Service List**, click **Cloud Eye** under **Management & Governance**.
- Step 3** In the navigation pane, choose **Cloud Service Monitoring > ModelArts**.
- Step 4** Click the down arrow next to the target real-time service name. Then, click **Create Alarm Rule** in the **Operation** column of the target version.
- Step 5** On the **Create Alarm Rule** page, create an alarm rule for model loads as prompted.

----End

Method 4: Setting an Alarm Rule for a Metric of a Service or Model Version

- Step 1** Log in to the management console.
- Step 2** In the **Service List**, click **Cloud Eye** under **Management & Governance**.
- Step 3** In the navigation pane, choose **Cloud Service Monitoring > ModelArts**.
- Step 4** Click the down arrow next to the target real-time service name. Then, click the target version and view alarm rule details.
- Step 5** On the alarm rule details page, click the plus sign (+) in the upper right corner of a metric and set an alarm rule for the metric.

----End

Viewing Monitoring Metrics

Cloud Eye on the cloud service platform monitors the statuses of ModelArts real-time services and model loads. You can obtain the monitoring metrics of each ModelArts real-time service and model on the management console. It takes a period of time to transmit and display monitored data. The statuses displayed on the Cloud Eye console are obtained 5 to 10 minutes before. You can view the monitoring data of a newly created real-time service 5 to 10 minutes later.

Prerequisites:

- The ModelArts real-time service is running properly.
- Alarm rules have been configured on the Cloud Eye page. For details, see [Setting Alarm Rules](#).

- The real-time service has been properly running for at least 10 minutes.
- The monitored data and graphics are available for a new real-time service after the service runs for at least 10 minutes.
- Cloud Eye does not display the metrics of a faulty or deleted real-time service. The monitoring metrics can be viewed after the real-time service starts or recovers.

The monitoring data is unavailable without alarm rules configured in Cloud Eye. For details, see [Setting Alarm Rules](#).

Step 1 Log in to the management console.

Step 2 In the **Service List**, click **Cloud Eye** under **Management & Governance**.

Step 3 In the navigation pane, choose **Cloud Service Monitoring > ModelArts**.

Step 4 View monitoring graphs.

- Viewing monitoring graphs of a real-time service: Click **View Metric** in the **Operation** column.
- Viewing monitoring graphs of the model loads: Click  next to the target real-time service, and click **View Metric** in the **Operation** column of the target model.

Step 5 In the monitoring area, you can select a period to view the monitoring data.

You can view the monitoring data in the last 1 hour, 3 hours, or 12 hours. To view the monitoring curve of a longer time range, click  to enlarge the graph.

-----End

8.7.6 Integrating a Real-Time Service API into the Production Environment

For a real-time service API that has been commissioned, you can integrate it into the production environment.

Prerequisites

The real-time service is running. Otherwise, applications in the production environment will be unavailable.

Integration Mode

ModelArts real-time services provide standard RESTful APIs, which can be accessed using HTTPS. ModelArts provides SDKs for calling real-time service APIs. For details about how to call the SDKs, see "Scenario 1: Perform an inference test using the predictor" in [SDK Reference](#).

In addition, you can use common development tools and languages to call the APIs. You can search for and obtain the guides for calling standard RESTful APIs on the Internet.

8.7.7 Configuring Auto Restart upon a Real-Time Service Fault

Description

When an Snt9b hardware fault is detected, the system automatically resets the Snt9B chip and restarts the real-time inference service. This helps to improve the recovery speed.

Constraints

Only synchronous real-time services using Snt9b resources are supported.

This function will reset the entire node. Before enabling it, ensure that the deployed real-time service is using specifications with 8 x N cards and properly evaluate the impacts on services running on the node.

Enabling Auto Restart

When deploying a real-time service, select **Configure Now** next to **Advanced Configuration** and enable auto restart.

8.8 Managing Batch Inference Jobs

8.8.1 Viewing Details About a Batch Service

After a model is deployed as a batch service, you can access the **Batch Services** page to view its details.

1. Log in to the ModelArts console and choose **Model Deployment > Batch Services**.
2. Click the name of the target service to access its details page.

View service information. For details, see [Table 8-39](#).

Table 8-39 Batch service parameters

Parameter	Description
Name	Name of the batch service.
Service ID	ID of the batch service.
Status	Status of the batch service.
Job ID	Job ID of the batch service.
Instance Flavor	Node flavor of the batch service.
Instances	Number of nodes of the batch service.
Start Time	Start time of the batch service job.

Parameter	Description
Environment Variable	Environment variables added during batch service creation.
End Time	End time of the batch service job.
Description	Service description, which you can click the edit button to modify.
Input Path	OBS path to the input data in the batch service.
Output Path	OBS path to the output data in the batch service.
Model Name & Version	Name and version of the model used by the batch service.
Advanced Log Management	This feature is disabled by default. The runtime logs of batch services are stored only in the ModelArts log system. If this feature is enabled, the runtime logs of batch services will be exported and stored in Log Tank Service (LTS). LTS automatically creates log groups and log streams and caches run logs generated within seven days by default. For details about LTS log management, see Log Tank Service. NOTE • This cannot be disabled once it is enabled. • You will be billed for log query and storage features provided by LTS. For details, see LTS Pricing Details. • Do not print unnecessary audio log files. Otherwise, system logs may fail to be displayed, and the error message "Failed to load audio" may be displayed.

3. Switch between tabs on the details page of a batch service to view more details. For details, see [Table 8-40](#).

Table 8-40 Batch service tabs

Parameter	Description
Events	This page displays key operations during service use, such as the service deployment progress, detailed causes of deployment exceptions, and time points when a service is started, stopped, or modified. Events are saved for one month and will be automatically cleared then. For details about how to view events of a service, see Viewing Events of a Real-Time Service.

Parameter	Description
Logs	This page displays the log information about each model in the service. You can view logs generated in the latest 5 minutes, latest 30 minutes, latest 1 hour, and user-defined time segment. You can select the start time and end time when defining a time segment. If this feature is enabled, the logs stored in LTS will be displayed. You can click View Complete Logs on LTS to view all logs. Log search rules: • Do not enter a string that contains any of the following delimiters: ,";=()[]{}@<>/: \n\t\r. • You can use exact search by keyword. A keyword refers to the word between two adjacent delimiters. • You can use fuzzy search by keyword. For example, you can enter error, er?or, rro*, or er*r. • You can enter a phrase for exact search. For example, Start to refresh. • Before enabling this feature, you can combine keywords with && or . For example, query logs&&erro* or query logs erro*. After enabling this feature, you can combine keywords with AND or OR. For example, query logs AND erro* or query logs OR erro*.

8.8.2 Viewing Events of a Batch Service

During the whole lifecycle of a service, every key event is automatically recorded. You can view the events on the details page of the service at any time.

This helps you better understand the service deployment and running process and accurately locate faults when a task exception occurs. You can check the following events:

Table 8-41 Events

Event Type	Event (xxx should be replaced with the actual value.)	Solution
Normal	The service starts to deploy.	N/A
Abnormal	Insufficient resources. Wait until idle resources are sufficient.	Wait until the resources are released and try again.

Event Type	Event (xxx should be replaced with the actual value.)	Solution
Abnormal	Insufficient xxx. The scheduling failed. Supplementary information: xxx	Learn about resource insufficiency details based on the supplementary information. For details, see FAQs .
Normal	The image starts to create.	N/A
Abnormal	Failed to create model image xxx. For details, see logs :/nxxx.	Locate and rectify the fault based on the build logs.
Abnormal	Failed to create the image.	Contact technical support.
Normal	The image is created.	N/A
Abnormal	Service xxx failed. Error: xxx	Locate and rectify the fault based on the error information.
Abnormal	Failed to update the service. Perform a rollback.	Contact technical support.
Normal	The service is being updated.	N/A
Normal	The service is being started.	N/A
Normal	The service is being stopped.	N/A
Normal	The service has been stopped.	N/A
Normal	Auto stop has been disabled.	N/A
Normal	Auto stop has been enabled. The service will stop after xs.	N/A
Normal	The service stops when the auto stop time expires.	N/A
Abnormal	The service is stopped because the quota exceeds the upper limit.	Contact technical support.

Event Type	Event (xxx should be replaced with the actual value.)	Solution
Abnormal	Failed to automatically stop the service. Error: xxx	Locate and rectify the fault based on the error information.
Normal	Service instances deleted from resource pool xxx.	N/A
Normal	Service instances stopped in resource pool xxx.	N/A
Abnormal	The batch service failed. Try again later. Error: xxx	Locate and rectify the fault based on the error information.
Normal	The service has been executed.	N/A
Abnormal	Failed to stop the service. Error: xxx	Locate and rectify the fault based on the error information.
Normal	The subscription license xxx is to expire.	N/A
Normal	Service xxx started.	N/A
Abnormal	Failed to start service xxx.	For details about how to locate and rectify the fault, see FAQs .
Abnormal	Service deployment timed out. Error: xxx	Locate and rectify the fault based on the error information.
Normal	Failed to update the service. The update has been rolled back.	N/A
Abnormal	Failed to update the service. The rollback failed.	Contact technical support.
Normal	[model 0.0.1] OBS bucket, OBS parallel file system, or SFS Turbo mounted successfully. [%s] %s volume successfully.	N/A

During service deployment and running, key events can both be manually and automatically refreshed.

Viewing Events

1. In the navigation pane of the ModelArts console, choose **Model Deployment > Batch Services**. In the service list, click the name or ID of the target service to go to its details page.
2. View the events in the **Events** tab.

8.8.3 Managing the Lifecycle of a Batch Service

Starting a Service

A service not in the **Deploying** state can be started. A service is billed from the time when it is running. To start a service, use either of the following methods:

- Log in to the ModelArts console and choose **Model Deployment** from the navigation pane. Go to the service management page of the target service. Click **Start** in the **Operation** column to start the target service.
- Log in to the ModelArts console and choose **Model Deployment** from the navigation pane. Go to the service management page of the target service. Click the name of the target service to access its details page. Click **Start** in the upper right corner of the page to start the service.

NOTE

Services deployed on ModelArts edge nodes or ModelArts edge resource pools cannot be started.

Stopping a Service

A stopped service will no longer be billed. Stop a service in either of the following ways:

- Log in to the ModelArts console and choose **Model Deployment > Batch Services**. Click **Stop** in the **Operation** column to stop the target service.
- Log in to the ModelArts console and choose **Model Deployment > Batch Services**. Click the name of the target service to access its details page. Click **Stop** in the upper right corner of the page to stop the service.

NOTE

Services deployed on ModelArts edge nodes or ModelArts edge resource pools cannot be stopped.

Deleting a Service

If a service is no longer in use, delete it to release resources.

Log in to the ModelArts console and choose **Model Deployment > Batch Services**.

- In the batch service list, click **Delete** in the **Operation** column of the target service to delete it.
- Select services in the batch service list and click **Delete** above the list to delete services in batches.
- Click the name of the target service. On the displayed service details page, click **Delete** in the upper right corner to delete the service.

 NOTE

- A deleted service cannot be recovered.
- A service cannot be deleted without agency authorization.

Restarting a Service

Batch services cannot be restarted.

8.8.4 Modifying a Batch Service

For a deployed service, you can modify its basic information to match service changes and change the model version to upgrade it.

You can modify the basic information about a service in either of the following ways:

Method 1: [Modify the Service Information on the Service Management Page](#)

Method 2: [Modify the Service Information on the Service Details Page](#)

Prerequisites

The service has been deployed. The service in the **Deploying** state cannot be upgraded by modifying the service information.

Constraints

- Improper upgrade operations will interrupt services during the upgrade.
- ModelArts supports hitless rolling upgrade of real-time services in some scenarios. Prepare for and fully verify the upgrade.

Table 8-42 Scenarios for hitless rolling upgrade

Meta Model Source	Using a Public Resource Pool	Using a Dedicated Resource Pool
Training job	Not supported	Not supported
Container image	Not supported	Supported. The custom image for creating a model must meet Specifications for Custom Images .
OBS	Not supported	Not supported

Method 1: Modify the Service Information on the Service Management Page

1. Log in to the ModelArts console and choose **Model Deployment** from the navigation pane. Go to the service management page of the target service.
2. In the service list, click **Modify** in the **Operation** column of the target service, modify basic service information, and submit the modification task as prompted.

When some parameters are modified, the system automatically restarts the service for the modification to take effect. When you submit a service modification task, if a restart is required, a dialog box will be displayed. For details about the batch service parameters, see [Deploying a Model as a Batch Inference Service](#).

Method 2: Modify the Service Information on the Service Details Page

1. Log in to the ModelArts console and choose **Model Deployment** from the navigation pane. Go to the service management page of the target service.
2. Click the name of the target service to access its details page.
3. Click **Modify** in the upper right corner of the page, modify the service details, and submit the modification task as prompted.

When some parameters are modified, the system automatically restarts the service for the modification to take effect. When you submit a service modification task, if a restart is required, a dialog box will be displayed. For details about the batch service parameters, see [Deploying a Model as a Batch Inference Service](#).

9 Creating a Custom Image for ModelArts Standard

9.1 Applications of Custom Images

During the development and runtime of AI services, complex environment dependencies need to be debugged for containerization. In the best practices of AI development in ModelArts, container images are used to provide fixed runtime environments. In this way, dependencies can be managed and the runtime environments can be easily switched. The container resources provided by ModelArts enable quick and efficient AI development and model experiment iteration.

This section describes the concepts of images, application scenarios of preset images and custom images, and how to create a custom image.

Concepts

Preset image: provided by ModelArts. You can select a ModelArts image when creating a notebook instance, training job, or model.

There are two types of preset images:

- **Unified image:** Use a unified image when you create a notebook instance, training job, or model. All images released later are unified images. For details, see [ModelArts Unified Images](#).
- **Unique image:** Use a unique image for a single module. For example, the preset image for training can only be used to create a training job. These images will be brought offline. For details, see [Preset Dedicated Images in Notebook Instances](#), [Preset Dedicated Images for Training](#), and [Preset Dedicated Images for Inference](#).

Custom image: created by user by following the ModelArts image creation specifications.

Base image: used for image creation. A base image can be a ModelArts preset image or a third-party image.

Custom Image Services

- SWR

Software Repository for Container (SWR) provides easy, secure, and reliable management over container images throughout their lifecycle, facilitating the deployment of containerized applications. You can upload, download, and manage container images through the SWR console, SWR APIs, or community CLI.

Your custom images must be uploaded to SWR. The custom images used by ModelArts for training or creating models are obtained from the SWR service management list.

Figure 9-1 Obtaining images



- OBS

Object Storage Service (OBS) is a cloud storage service optimized for storing massive amounts of data. It provides unlimited, secure, and highly reliable storage capabilities at a relatively low cost.

ModelArts exchanges data with OBS. You can store data in OBS.

- ECS

An Elastic Cloud Server (ECS) is a basic computing unit that consists of vCPUs, memory, OS, and Elastic Volume Service (EVS) disks. After an ECS is created, you can use it as your local PC or physical server.

You can create a custom image on premises or on an ECS.

NOTE

When you use a custom image, ModelArts may need to access dependent services, such as SWR and OBS. The custom image can be used only after the access is authorized. It is a good practice to use an agency for authorization. After an agency is configured, the permissions to access dependent services are delegated to ModelArts so that ModelArts can use the dependent services and perform operations on resources on your behalf. For details, see [Configuring Agency Authorization for ModelArts with One Click](#).

Application Scenarios of Preset Images

ModelArts provides a group of preset images. You can use a preset image to create a notebook instance. After installing and configuring dependencies on the instance, create a custom image. Then, you can directly use the image in ModelArts for training jobs without any adaptation. You can also use preset images to submit training jobs and create models.

We recommend the preset image version based on your development requirements and stability of the version. If your development can be carried out using versions preset in ModelArts, for example, MindSpore 1.X, use the preset images. They have been fully verified and have many commonly-used installation packages, relieving you from configuring the environment.

The preset images provided by ModelArts by default have the following features:

- Out-of-the-box and scenario-specific: Typical dependent environments for AI development are preset in these images to provide optimal software, OS, and network configurations. They have been fully tested on hardware to ensure optimal compatibility and performance.
- Configuration customizable: Preset images are stored in the SWR repository for you to customize and register them as your own images.
- Secure and reliable: Access policies, user permissions control, vulnerability scanning for development software, and OS are configured based on best practices for security hardening to ensure the security of images.

Application Scenarios of Custom Images

Preset images cannot meet the requirements for deep learning engines and development library. To resolve this issue, ModelArts allows image customization so that you can customize runtime engines.

ModelArts runs in containers. You can customize container images to run on ModelArts. Custom images support CLI parameters and environment variables in free-text format, featuring high flexibility for a wide range of compute engines.

When you create a custom image, use the ModelArts preset image as the base image, that is, obtain the preset image using the SWR address in Dockerfile for image creation. You can obtain the SWR address of the image in the ModelArts preset image list. For details, see [Preset Images Supported by ModelArts](#) and .

- **Creating a Custom Image for a Notebook Instance**
If the preset images of notebook instances cannot meet requirements, you can create a custom image by installing and configuring the software and other data required by the environment in a preset image. Then, use the custom image to create new notebook instances. You can also create a custom image based on existing images in notebook instances.
- **Creating a Custom Image for Model Training**
If you have developed a model or training script locally but the AI engine you used is not supported by ModelArts, create a custom image and upload it to SWR. Then, use this image to create a training job on ModelArts and use the resources provided by ModelArts to train models.
- **Creating a Custom Image for Inference**
If you have developed a model using an AI engine that is not supported by ModelArts, to use this model to create AI applications, create a custom image, import the image to ModelArts, and use it to create models. The models created in this way can be centrally managed and deployed as services.

9.2 Preset Images Supported by ModelArts

9.2.1 ModelArts Preset Image Updates

This section describes the updates of ModelArts preset images, including changes of dependencies, so that you can learn about the changes and use images more easily.

Unified Image Updates

Table 9-1 Unified image updates

Image	Updated	Description
mindspore_2.3.0-cann_8.0.rc1-py_3.9-euler_2.10.7-aarch64-snt9b	2024-05-21	Based on the commercial version of Ascend 415, MindSpore is updated to 2.3.0-rc4, and CANN is updated to 8.0.rc1. Bring ma-cau 1.1.6 and ma-cau-adapter 1.1.3 offline.
pytorch_2.1.0-cann_8.0.rc1-py_3.9-euler_2.10.7-aarch64-snt9b	2024-05-21	Based on the commercial version of Ascend 415, CANN is updated to 8.0.rc1.
pytorch_1.11.0-cann_8.0.rc1-py_3.9-euler_2.10.7-aarch64-snt9b	2024-05-21	Based on the commercial version of Ascend 415, CANN is updated to 8.0.rc1.
mindspore_2.3.0-cann_8.0.rc2-py_3.9-euler_2.10.7-aarch64-snt9b	2024-07-27	Based on the commercial version of Ascend 715, MindSpore is updated to 2.3.0, CANN is updated to 8.0.rc2, and the driver is Ascend HDK 24.1.RC2.
pytorch_2.1.0-cann_8.0.rc2-py_3.9-euler_2.10.7-aarch64-snt9b	2024-07-27	Based on the commercial version of Ascend 715, CANN is updated to 8.0.rc2, and the driver is Ascend HDK 24.1.RC2.
pytorch_1.11.0-cann_8.0.rc2-py_3.9-euler_2.10.7-aarch64-snt9b	2024-07-27	Based on the commercial version of Ascend 715, CANN is updated to 8.0.rc2, and the driver is Ascend HDK 24.1.RC2.
mindspore_2.4.0-cann_8.0.rc3-py_3.9-euler_2.10.10-aarch64-snt9b	2024-11-08	Based on the commercial version of Ascend 930, CANN is updated to 8.0.rc3, and the driver is Ascend HDK 24.1.RC3.
pytorch_2.1.0-cann_8.0.rc3-py_3.9-euler_2.10.10-aarch64-snt9b	2024-11-15	Based on the commercial version of Ascend 930, CANN is updated to 8.0.rc3, and the driver is Ascend HDK 24.1.RC3.
pytorch_2.1.0-cann_8.0.0-py_3.10-euler_2.10.11-aarch64-snt9b	2025-03-05	Based on the commercial version of Ascend TR6, CANN is updated to 8.0.0.beta1, and the driver is Ascend HDK 24.1.0.

Image	Updated	Description
mindspore_2.4.10-cann_8.0.0-py_3.10-euler_2.10.11-aarch64-snt9b	2025-03-05	Based on the commercial version of Ascend TR6, MindSpore is updated to 2.4.10, CANN is updated to 8.0.0.beta1, and the driver is Ascend HDK 24.1.0.
pytorch_2.1.0-cann_8.0.0-py_3.10-euler_2.10.11-aarch64-snt9	2025-03-05	Based on the commercial version of Ascend TR6, CANN is updated to 8.0.0.beta1, and the driver is Ascend HDK 24.1.0.
mindspore_2.4.10-cann_8.0.0-py_3.10-euler_2.10.11-aarch64-snt9	2025-03-05	Based on the commercial version of Ascend TR6, MindSpore is updated to 2.4.10, CANN is updated to 8.0.0.beta1, and the driver is Ascend HDK 24.1.0.
pytorch_2.1.0-cann_8.1.rc1-py_3.10-euler_2.10.11-aarch64-snt9b	2025-06-10	Based on the commercial version of Ascend 430, CANN is updated to 8.1.RC1.B150, and the driver is Ascend HDK 25.0.RC1.
mindspore_2.6.0rc1-cann_8.1.rc1-py_3.10-euler_2.10.11-aarch64-snt9b	2025-06-10	Based on the commercial version of Ascend 430, MindSpore is updated to 2.6.0rc1, CANN is updated to 8.1.RC1.B150, and the driver is Ascend HDK 25.0.RC1.
pytorch_2.1.0-cann_8.1.rc1-py_3.10-hce_2.0.2503-aarch64-snt3p	2025-06-10	Based on the commercial version of Ascend 430, MindSpore is updated to 2.6.0rc1, CANN is updated to 8.1.RC1.B150, and the driver is Ascend HDK 25.0.RC1.
mindspore_2.6.0rc1-cann_8.1.rc1-py_3.10-hce_2.0.2503-aarch64-snt3p	2025-06-10	Based on the commercial version of Ascend 430, CANN is updated to 8.1.RC1.B150, and the driver is Ascend HDK 25.0.RC1.

9.2.2 ModelArts Unified Images

Unified Image List

ModelArts provides unified images of Arm+Ascend specifications, including MindSpore and PyTorch. You can use the images to develop environment, train models, and deploy services. For details, see [Unified Image List](#).

Table 9-2 MindSpore

Preset Image	Supported Processor	Applicable Scope
mindspore_2.2.0-cann_7.0.1-py_3.9-euler_2.10.7-aarch64-snt9b	Ascend snt9b	Notebook, training, and inference deployment
mindspore_2.3.0-cann_8.0.rc2-py_3.9-euler_2.10.7-aarch64-snt9b	Ascend snt9b	Notebook, training, and inference deployment
mindspore_2.4.0-cann_8.0.rc3-py_3.9-euler_2.10.10-aarch64-snt9b	Ascend snt9b	Notebook, training, and inference deployment
mindspore_2.4.10-cann_8.0.0-py_3.10-euler_2.10.11-aarch64-snt9b	Ascend snt9b	Notebook, training, and inference deployment
mindspore_2.4.10-cann_8.0.0-py_3.10-euler_2.10.11-aarch64-snt9	Ascend snt9	Notebook, training, and inference deployment
mindspore_2.6.0rc1-cann_8.1.rc1-py_3.10-euler_2.10.11-aarch64-snt9b	Ascend snt9b	Notebook, training, and inference deployment
mindspore_2.6.0rc1-cann_8.1.rc1-py_3.10-hce_2.0.2503-aarch64-snt3p	Ascend snt3p	Inference deployment

Table 9-3 PyTorch

Preset Image	Supported Processor	Applicable Scope
pytorch_2.1.0-cann_7.0.1-py_3.9-euler_2.10.7-aarch64-snt9b	Ascend snt9b	Notebook, training, and inference deployment
pytorch_1.11.0-cann_7.0.1-py_3.9-euler_2.10.7-aarch64-snt9b	Ascend snt9b	Notebook, training, and inference deployment
pytorch_2.1.0-cann_8.0.rc2-py_3.9-euler_2.10.7-aarch64-snt9b	Ascend snt9b	Notebook, training, and inference deployment
pytorch_1.11.0-cann_8.0.rc2-py_3.9-euler_2.10.7-aarch64-snt9b	Ascend snt9b	Notebook, training, and inference deployment
pytorch_2.1.0-cann_8.0.rc3-py_3.9-euler_2.10.10-aarch64-snt9b	Ascend snt9b	Notebook, training, and inference deployment
pytorch_2.1.0-cann_8.0.0-py_3.10-euler_2.10.11-aarch64-snt9b	Ascend snt9b	Notebook, training, and inference deployment
pytorch_2.1.0-cann_8.0.0-py_3.10-euler_2.10.11-aarch64-snt9	Ascend snt9	Notebook, training, and inference deployment

Preset Image	Supported Processor	Applicable Scope
pytorch_2.1.0-cann_8.1.rc1-py_3.10-euler_2.10.11-aarch64-snt9b	Ascend snt9b	Notebook, training, and inference deployment
pytorch_2.1.0-cann_8.1.rc1-py_3.10-hce_2.0.2503-aarch64-snt3p	Ascend snt3p	Inference deployment

mindspore_2.6.0rc1-cann_8.1.rc1-py_3.10-hce_2.0.2503-aarch64-snt3p

Table 9-4 Information about the image

AI Engine	URL	Dependency	
mindspore 2.6.0rc1 + mindspore-lite 2.6.0rc1 + Ascend CANN Toolkit 8.1.RC1.B150	swr.<region>.myhuaweicloud.com/atelier/ mindspore_ascend:mindspore_2.6.0rc1-cann_8.1.rc1-py_3.10-hce_2.0.2503-aarch64-snt3p-20250603204733-091164e Example: CN-Hong Kong swr.cn-ap-southeast-1.myhuaweicloud.com/atelier/ mindspore_ascend:mindspore_2.6.0rc1-cann_8.1.rc1-py_3.10-hce_2.0.2503-aarch64-snt3p-20250603204733-091164e	PyPI package	YUM package

AI Engine	URL	Dependency	
		mindspore 2.4.10	cmake
		mindspore-lite 2.4.10	cpp
		mindinsight 2.3.0	curl
		mindarmour 2.0.0	ffmpeg
		mindformers 1.3.2	g++
		mindx-elastic 0.0.1	gcc
		seccomponent 1.1.8	git
		moxing-framework 2.3.3	grep
		ipykernel 6.7.0	python3
		ipython 8.18.1	rpm
		jupyter-client 7.4.9	tar
		matplotlib 3.7.3	unzip
		numpy 1.23.0	wget
		pandas 1.3.5	zip
		Pillow 10.3.0	
		pip 23.1.2	
		psutil 6.0.0	
		PyYAML 6.0.2	
		scipy 1.12.0	
		scikit-learn 1.5.1	
		tornado 6.4.1	
		msit 7.0.0rc1230	
		msit-compare 7.0.0rc1230	

mindspore_2.6.0rc1-cann_8.1.rc1-py_3.10-euler_2.10.11-aarch64-snt9b**Table 9-5** Information about the image

AI Engine	URL	Dependency	
		PyPI package	YUM package
mindspore 2.6.0rc1 + mindspore-lite 2.6.0rc1 + Ascend CANN Toolkit 8.1.RC1.B150	swr.<region>.myhuaweicloud.com/atelier/mindspore_ascend:mindspore_2.6.0rc1-cann_8.1.rc1-py_3.10-euler_2.10.11-aarch64-snt9b-20250628184338-b1da9c0 Example: CN-Hong Kong swr.cn-ap-southeast-1.myhuaweicloud.com/atelier/mindspore_ascend:mindspore_2.6.0rc1-cann_8.1.rc1-py_3.10-euler_2.10.11-aarch64-snt9b-20250628184338-b1da9c0	mindspore 2.4.10 mindspore-lite 2.4.10 mindinsight 2.3.0 mindarmour 2.0.0 mindformers 1.3.2 mindx-elastic 0.0.1 seccomponent 1.1.8 moxing-framework 2.3.3 ipykernel 6.7.0 ipython 8.18.1 jupyter-client 7.4.9 matplotlib 3.7.3 numpy 1.23.0 pandas 1.3.5 Pillow 10.3.0 pip 23.1.2 psutil 6.0.0 PyYAML 6.0.2 scipy 1.12.0 scikit-learn 1.5.1 tornado 6.4.1 msit 7.0.0rc1230 msit-compare 7.0.0rc1230	cmake cpp curl ffmpeg g++ gcc git grep python3 rpm tar unzip wget zip

mindspore_2.4.10-cann_8.0.0-py_3.10-euler_2.10.11-aarch64-snt9b

Table 9-6 Information about the image

AI Engine	URL	Dependency	
mindspore 2.4.10 + mindspore-lite 2.4.10 + Ascend CANN Toolkit 8.0.0	swr.<region>.myhuaweicloud.com/atelier/ mindspore_2_4_ascend:mindspore_2.4.10-cann_8.0.0-py_3.10-euler_2.10.11-aarch64-snt9b-20250305173557-cb53968 Example: CN-Hong Kong swr.cn-ap-southeast-1.myhuaweicloud.com/atelier/ mindspore_2_4_ascend:mindspore_2.4.10-cann_8.0.0-py_3.10-euler_2.10.11-aarch64-snt9b-20250305173557-cb53968	PyPI package	YUM package
		mindspore 2.4.10 mindspore-lite 2.4.10 mindinsight 2.3.0 mindarmour 2.0.0 mindformers 1.3.2 mindx-elastic 0.0.1 seccomponent 1.1.8 moxing-framework 2.2.10 ipykernel 6.7.0 ipython 8.18.1 jupyter-client 7.4.9 matplotlib 3.7.3 numpy 1.23.0 pandas 1.3.5 Pillow 10.3.0 pip 23.1.2 psutil 6.0.0 PyYAML 6.0.2 scipy 1.12.0 scikit-learn 1.5.1 tornado 6.4.1 msit 7.0.0rc1230 msit-compare 7.0.0rc1230	cmake cpp curl ffmpeg g++ gcc git grep python3 rpm tar unzip wget zip

mindspore_2.4.10-cann_8.0.0-py_3.10-euler_2.10.11-aarch64-snt9

Table 9-7 Information about the image

AI Engine	URL	Dependency	
		PyPI package	YUM package
mindspore 2.4.10 + mindspore-lite 2.4.10 + Ascend CANN Toolkit 8.0.0	swr.<region>.myhuaweicloud.com/atelier/ mindspore_2_4_ascend:mindspore_2.4.10-cann_8.0.0-py_3.10-euler_2.10.11-aarch64-snt9-20250305173557-cb53968 Example: CN-Hong Kong swr.cn-ap-southeast-1.myhuaweicloud.com/atelier/ mindspore_2_4_ascend:mindspore_2.4.10-cann_8.0.0-py_3.10-euler_2.10.11-aarch64-snt9-20250305173557-cb53968	mindspore 2.4.10 mindspore-lite 2.4.10 mindinsight 2.3.0 mindarmour 2.0.0 mindformers 1.3.2 mindx-elastic 0.0.1 seccomponent 1.1.8 moxing-framework 2.2.10 ipykernel 6.7.0 ipython 8.18.1 jupyter-client 7.4.9 matplotlib 3.7.3 numpy 1.23.0 pandas 1.3.5 Pillow 10.3.0 pip 23.1.2 psutil 6.0.0 PyYAML 6.0.2 scipy 1.12.0 scikit-learn 1.5.1 tornado 6.4.1 msit 7.0.0rc1230 msit-compare 7.0.0rc1230	cmake cpp curl ffmpeg g++ gcc git grep python3 rpm tar unzip wget zip

mindspore_2.4.0-cann_8.0.rc3-py_3.9-euler_2.10.10-aarch64-snt9b**Table 9-8** Information about the image

AI Engine	URL	Dependency	
		PyPI package	YUM package
mindspore 2.4.0 + mindspore-lite 2.4.0 + Ascend CANN Toolkit 8.0.rc3	swr.<region>.myhuaweicloud.com/atelier/ mindspore_2_4_ascend:mindspore_2.4.0-cann_8.0.rc3-py_3.9-euler_2.10.10-aarch64-snt9b-20241108112232-e95a4d0 Example: CN-Hong Kong swr.cn-ap-southeast-1.myhuaweicloud.com/atelier/ mindspore_2_4_ascend:mindspore_2.4.0-cann_8.0.rc3-py_3.9-euler_2.10.10-aarch64-snt9b-20241108112232-e95a4d0	mindspore 2.4.0 mindspore-lite 2.4.0 mindinsight 2.3.0 mindarmour 2.0.0 mindformers 1.3.0 mindx-elastic 0.0.1 seccomponent 1.1.8 moxing-framework 2.2.10 ipykernel 6.7.0 ipython 8.18.1 jupyter-client 7.4.9 matplotlib 3.7.3 numpy 1.23.0 pandas 1.3.5 Pillow 10.3.0 pip 21.0.1 psutil 6.0.0 PyYAML 6.0.2 scipy 1.12.0 scikit-learn 1.5.1 tornado 6.4.1	cmake cpp curl ffmpeg g++ gcc git grep python3 rpm tar unzip wget zip

mindspore_2.3.0-cann_8.0.rc2-py_3.9-euler_2.10.7-aarch64-snt9b

Table 9-9 Information about the image

AI Engine	URL	Dependency	
mindspore 2.3.0 + mindspore-lite 2.3.0 + Ascend CANN Toolkit 8.0.rc2	swr.<region>.myhuaweicloud.com/atelier/ mindspore_2_3_ascend:mindspore_2.3.0-cann_8.0.rc2-py_3.9-euler_2.10.7-aarch64-snt9b-20240727152329-0f2c29a Example: CN-Hong Kong swr.cn-ap-southeast-1.myhuaweicloud.com/atelier/ mindspore_2_3_ascend:mindspore_2.3.0-cann_8.0.rc2-py_3.9-euler_2.10.7-aarch64-snt9b-20240727152329-0f2c29a	PyPI package	YUM package
		mindspore 2.3.0 mindspore-lite 2.3.0 mindinsight 2.3.0 mindarmour 2.0.0 mindformers 1.2.0 seccomponent 1.1.8 moxing-framework 2.2.8.0aa484aa ipykernel 6.7.0 ipython 8.18.1 jupyter-client 7.4.9 matplotlib 3.5.1 numpy 1.22.0 pandas 1.3.5 Pillow 10.0.1 pip 21.0.1 psutil 5.9.5 PyYAML 6.0.1 scipy 1.10.1 scikit-learn 1.0.2 tornado 6.4	cmake cpp curl ffmpeg g++ gcc git grep python3 rpm tar unzip wget zip

pytorch_2.1.0-cann_8.1.rc1-py_3.10-hce_2.0.2503-aarch64-snt3p**Table 9-10** Information about the image

AI Engine	URL	Dependency	
		PyPI package	YUM package
pytorch 2.1.0 + Ascend CANN Toolkit 8.1.RC1.B150	swr.<region>.myhuaweicloud.com/atelier/pytorch_2_1_ascend:pytorch_2.1.0-cann_8.1.rc1-py_3.10-hce_2.0.2503-aarch64-snt3p-20250603154214-4e60e43	torch 2.1.0 torch-npu 2.1.0.post10 seccomponent 1.1.8 moxing-framework 2.3.3 ipykernel 6.7.0 ipython 8.18.1 jupyter-client 7.4.9 ma-cau 1.1.7 ma-cau-adapter 1.1.3 ma-cli 1.2.3 matplotlib 3.7.3 numpy 1.23.0 pandas 1.3.5 Pillow 10.3.0 pip 23.1.2 psutil 6.0.0 PyYAML 6.0.2 scipy 1.12.0 scikit-learn 1.5.1 tornado 6.4.1 msit 7.0.0rc1230 msit-llm 7.0.0rc1230	cmake cpp curl ffmpeg g++ gcc git grep python3 rpm tar unzip wget zip

pytorch_2.1.0-cann_8.1.rc1-py_3.10-euler_2.10.11-aarch64-snt9b**Table 9-11** Information about the image

AI Engine	URL	Dependency	
		PyPI package	YUM package
pytorch 2.1.0 + Ascend CANN Toolkit 8.1.RC1.B150	swr.<region>.myhuaweicloud.com/atelier/pytorch_2.1_ascend:pytorch_2.1.0-cann_8.1.rc1-py_3.10-euler_2.10.11-aarch64-snt9b-20250603154214-4e60e43	torch 2.1.0 torch-npu 2.1.0.post10 seccomponent 1.1.8 moxing-framework 2.3.3 ipykernel 6.7.0 ipython 8.18.1 jupyter-client 7.4.9 ma-cau 1.1.7 ma-cau-adapter 1.1.3 ma-cli 1.2.3 matplotlib 3.7.3 numpy 1.23.0 pandas 1.3.5 Pillow 10.3.0 pip 23.1.2 psutil 6.0.0 PyYAML 6.0.2 scipy 1.12.0 scikit-learn 1.5.1 tornado 6.4.1 msit 7.0.0rc1230 msit-llm 7.0.0rc1230	cmake cpp curl ffmpeg g++ gcc git grep python3 rpm tar unzip wget zip

pytorch_2.1.0-cann_8.0.0-py_3.10-euler_2.10.11-aarch64-snt9b

Table 9-12 Information about the image

AI Engine	URL	Dependency	
		PyPI package	YUM package
pytorch 2.1.0 + mindsore-lite 2.4.0 + Ascend CANN Toolkit 8.0.rc3	swr.<region>.myhuaweicloud.com/atelier/pytorch_2.1_ascend:pytorch_2.1.0-cann_8.0.0-py_3.10-euler_2.10.11-aarch64-snt9b-20250305173557-cb53968	torch 2.1.0 torch-npu 2.1.0.post10 seccomponent 1.1.8 moxing-framework 2.2.10 ipykernel 6.7.0 ipython 8.18.1 jupyter-client 7.4.9 ma-cau 1.1.7 ma-cau-adapter 1.1.3 ma-cli 1.2.3 matplotlib 3.7.3 numpy 1.23.0 pandas 1.3.5 Pillow 10.3.0 pip 23.1.2 psutil 6.0.0 PyYAML 6.0.2 scipy 1.12.0 scikit-learn 1.5.1 tornado 6.4.1 msit 7.0.0rc1230 msit-llm 7.0.0rc1230	cmake cpp curl ffmpeg g++ gcc git grep python3 rpm tar unzip wget zip

pytorch_2.1.0-cann_8.0.0-py_3.10-euler_2.10.11-aarch64-snt9**Table 9-13** Information about the image

AI Engine	URL	Dependency	
		PyPI package	YUM package
pytorch 2.1.0 + mindsore-lite 2.4.0 + Ascend CANN Toolkit 8.0.rc3	swr.<region>.myhuaweicloud.com/atelier/pytorch_2.1_ascend:pytorch_2.1.0-cann_8.0.0-py_3.10-euler_2.10.11-aarch64-snt9-20250305173557-cb53968	torch 2.1.0 torch-npu 2.1.0.post10 seccomponent 1.1.8 moxing-framework 2.2.10 ipykernel 6.7.0 ipython 8.18.1 jupyter-client 7.4.9 ma-cau 1.1.7 ma-cau-adapter 1.1.3 ma-cli 1.2.3 matplotlib 3.7.3 numpy 1.23.0 pandas 1.3.5 Pillow 10.3.0 pip 23.1.2 psutil 6.0.0 PyYAML 6.0.2 scipy 1.12.0 scikit-learn 1.5.1 tornado 6.4.1 msit 7.0.0rc1230 msit-llm 7.0.0rc1230	cmake cpp curl ffmpeg g++ gcc git grep python3 rpm tar unzip wget zip

pytorch_2.1.0-cann_8.0.rc3-py_3.9-euler_2.10.10-aarch64-snt9b**Table 9-14** Information about the image

AI Engine	URL	Dependency	
		PyPI package	YUM package
pytorch 2.1.0 + mindspore-lite 2.4.0 + Ascend CANN Toolkit 8.0.rc3	swr.<region>.myhuaweicloud.com/atelier/pytorch_2_1_ascend:pytorch_2.1.0-cann_8.0.rc3-py_3.9-euler_2.10.10-aarch64-snt9b-20241115095809-5702f8a pytorch_2.1.0-cann_8.0.rc3-py_3.9-euler_2.10.10-aarch64-snt9b-20241115095809-5702f8a	torch 2.1.0 torch-npu 2.1.0.post8.dev20241029 mindspore-lite 2.4.0 seccomponent 1.1.8 moxing-framework 2.2.10 ipykernel 6.7.0 ipython 8.18.1 jupyter-client 7.4.9 ma-cau 1.1.7 ma-cau-adapter 1.1.3 ma-cli 1.2.3 matplotlib 3.7.3 numpy 1.23.0 pandas 1.3.5 Pillow 10.3.0 pip 21.0.1 psutil 6.0.0 PyYAML 6.0.2 scipy 1.12.0 scikit-learn 1.5.1 tornado 6.4.1	cmake cpp curl ffmpeg g++ gcc git grep python3 rpm tar unzip wget zip

pytorch_2.1.0-cann_8.0.rc2-py_3.9-euler_2.10.7-aarch64-snt9b

Table 9-15 Information about the image

AI Engine	URL	Dependency	
		PyPI package	YUM package
pytorch 2.1.0 + mindspore-lite 2.3.0 + Ascend CANN Toolkit 8.0.rc2	swr.<region>.myhuaweicloud.com/atelier/pytorch_2.1.0-ascend:pytorch_2.1.0-cann_8.0.rc2-py_3.9-euler_2.10.7-aarch64-snt9b-20240727152329-0f2c2	torch 2.1.0 torch-npu 2.1.0.post6.dev20240716 mindspore-lite 2.3.0 seccomponent 1.1.8 moxing-framework 2.2.8.0aa484aa ipykernel 6.7.0 ipython 8.18.1 jupyter-client 7.4.9 ma-cau 1.1.7 ma-cau-adapter 1.1.3 ma-cli 1.2.3 matplotlib 3.5.1 numpy 1.22.0 pandas 1.3.5 Pillow 10.0.1 pip 21.0.1 psutil 5.9.5 PyYAML 6.0.1 scipy 1.10.1 scikit-learn 1.0.2 tornado 6.4	cmake cpp curl ffmpeg g++ gcc git grep python3 rpm tar unzip wget zip

pytorch_1.11.0-cann_8.0.rc2-py_3.9-euler_2.10.7-aarch64-snt9b**Table 9-16** Information about the image

AI Engine	URL	Dependency	
		PyPI package	YUM package
pytorch 1.11 + mindspore-lite 2.3.0 + Ascend CANN Toolkit 8.0.rc2	swr.<region>.myhuaweicloud.com/atelier/pytorch_1_11_ascend:pytorch_1.11.0-cann_8.0.rc2-py_3.9-euler_2.10.7-aarch64-snt9b-20240727152329-0f2c29a	torch 1.11.0 torch-npu 1.11.0.post14.dev20240716 apex 0.1.dev20240716+ascend mindspore-lite 2.3.0 seccomponent 1.1.8 moxing-framework 2.2.8.0aa484aa ipykernel 6.7.0 ipython 8.18.1 jupyter-client 7.4.9 ma-cau 1.1.7 ma-cau-adapter 1.1.3 ma-cli 1.2.3 matplotlib 3.5.1 numpy 1.22.0 pandas 1.3.5 Pillow 10.3.0 pip 21.0.1 psutil 5.9.5 PyYAML 6.0.1 scipy 1.10.1 scikit-learn 1.0.2 tornado 6.4	cmake cpp curl ffmpeg g++ gcc git grep python3 rpm tar unzip wget zip

mindspore_2.2.0-cann_7.0.1-py_3.9-euler_2.10.7-aarch64-snt9b**Table 9-17** Information about the image

AI Engine	URL	Dependency	
		PyPI package	YUM package
mindspore 2.2.0 + mindspore-lite 2.2.0 + Ascend CANN Toolkit 7.0.RC1	swr.<region>.myhuaweicloud.com/atelier/ mindspore_2_2_ascend:mindspore_2.2.0-cann_7.0.1-py_3.9-euler_2.10.7-aarch64-snt9b-20231107190844-50a1a83 Example: CN-Hong Kong swr.cn-ap-southeast-1.myhuaweicloud.com/atelier/ mindspore_2_2_ascend:mindspore_2.2.0-cann_7.0.1-py_3.9-euler_2.10.7-aarch64-snt9b-20231107190844-50a1a83	mindspore 2.2.0 ipykernel 6.7.0 ipython 8.18.1 jupyter-client 7.4.9 ma-cli 1.2.3 matplotlib 3.5.1 modelarts 1.4.20 moxing-framework 2.2.3.2c7f2141 numpy 1.22.0 pandas 1.3.5 Pillow 10.0.1 pip 21.0.1 psutil 5.9.5 PyYAML 6.0.1 scipy 1.10.1 scikit-learn 1.0.2 tornado 6.4 mindinsight 2.2.0	cmake cpp curl ffmpeg g++ gcc git grep python3 rpm tar unzip wget zip

pytorch_2.1.0-cann_7.0.1-py_3.9-euler_2.10.7-aarch64-snt9b

Table 9-18 Information about the image

AI Engine	URL	Dependency	
pytorch 2.1.0 + mindsore-lite 2.2.0 + Ascend CANN Toolkit 7.0.RC1	swr.<region>.myhuaweicloud.com/atelier/pytorch_2_1_ascend:pytorch_2.1.0-cann_7.0.1-py_3.9-euler_2.10.7-aarch64-snt9b-20231107190844-50a1a83 Example: CN-Hong Kong swr.cn-ap-southeast-1.myhuaweicloud.com/atelier/pytorch_2_1_ascend:pytorch_2.1.0-cann_7.0.1-py_3.9-euler_2.10.7-aarch64-snt9b-20231107190844-50a1a83	PyPI package	YUM package
		torch 2.1.0 apex 0.1-ascend-20231013 torch-npu 2.1.0rc1.post20231013 ipykernel 5.3.4 ipython 7.34.0 jupyter-client 7.3.4 ma-cli 1.2.3 matplotlib 3.5.1 modelarts 1.4.26 moxing-framework 2.2.3.2c7f2141 numpy 1.26.1 pandas 1.3.5 Pillow 10.0.1 pip 21.0.1 psutil 5.9.5 PyYAML 6.0.1 scipy 1.10.1 scikit-learn 1.0.2 tornado 6.3.3	cmake cpp curl ffmpeg g++ gcc git grep python3 rpm tar unzip wget zip

pytorch_1.11.0-cann_7.0.1-py_3.9-euler_2.10.7-aarch64-snt9b**Table 9-19** Information about the image

AI Engine	URL	Dependency	
pytorch 1.11 + mindspore-lite 2.2.0 + Ascend CANN Toolkit 7.0.RC1	swr.<region>.myhuaweicloud.com/atelier/pytorch_1_11_ascend:pytorch_1.11.0-cann_7.0.1-py_3.9-euler_2.10.7-aarch64-snt9b-20231107190844-50a1a83 Example: CN-Hong Kong swr.cn-ap-southeast-1.myhuaweicloud.com/atelier/pytorch_1_11_ascend:pytorch_1.11.0-cann_7.0.1-py_3.9-euler_2.10.7-aarch64-snt9b-20231107190844-50a1a83	PyPI package	YUM package
		torch 1.11.0 torch-npu 1.11.0.post4-20231013 apex 0.1-ascend-20231013 ipynb 6.7.0 ipython 8.17.2 jupyter-client 7.4.9 ma-cli 1.2.3 matplotlib 3.5.1 modelarts 1.4.20 moxing-framework 2.2.3.2c7f2141 numpy 1.26.1 pandas 1.3.5 Pillow 10.0.1 pip 21.0.1 psutil 5.9.5 PyYAML 6.0.1 scipy 1.10.1 scikit-learn 1.0.2 tornado 6.3.3	cmake cpp curl ffmpeg g++ gcc git grep python3 rpm tar unzip wget zip

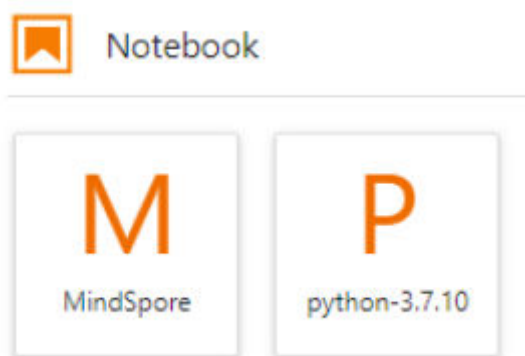
9.2.3 Preset Dedicated Images in Notebook Instances

ModelArts DevEnviron provides Docker container images, which can run as preset containers. Certain preset images are built on common AI engine frameworks such as PyTorch, TensorFlow, and MindSpore. These images are named using the AI engines. Additionally, many common packages are preset in these images, relieving you from the package installation.

The images preset in ModelArts DevEnviron include:

- Typical preset packages: AI engines based on standard Conda, data analysis software packages such as Pandas and Numpy, and tool software such as CUDA and cuDNN are included to meet your needs.

- Preset Conda environments: A Conda environment and basic Conda Python (excluding any AI engine) are created for each preset image. The following figure shows the Conda environment for a preset MindSpore image.



Select a Conda environment based on whether MindSpore is used for debugging.

- Notebook: a web application that enables you to code on the GUI and combine the code, mathematical equations, and visualized content into a document.
- JupyterLab plug-ins: enable flavor changing, case sharing to AI Gallery for communication, and instance stopping to improve user experience.
- Remote SSH: allows you to remotely start and debug a notebook instance from a local PC.
- Images preset in ModelArts DevEnviron: After these preset images support function development, the custom images created based on these preset images can be directly used for ModelArts training jobs.

Table 9-20 Preset x86 images

Engine	Image
PyTorch	pytorch_2.1.0-cuda_12.1-py_3.10.6-ubuntu_22.04-x86_64
	pytorch1.8-cuda10.2-cudnn7-ubuntu18.04
	pytorch1.10-cuda10.2-cudnn7-ubuntu18.04
	pytorch1.4-cuda10.1-cudnn7-ubuntu18.04
Tensorflow	tensorflow2.1-cuda10.1-cudnn7-ubuntu18.04
	tensorflow1.13-cuda10.0-cudnn7-ubuntu18.04
MindSpore	mindspore1.7.0-cuda10.1-py3.7-ubuntu18.04
	mindspore1.7.0-py3.7-ubuntu18.04
	mindspore1.2.0-cuda10.1-cudnn7-ubuntu18.04
	mindspore1.2.0-openmpi2.1.1-ubuntu18.04

Engine	Image
No AI engine (base images dedicated for image customization)	conda3-cuda10.2-cudnn7-ubuntu18.04
	conda3-ubuntu18.04

x86-powered PyTorch Base Images

PyTorch contains three types of images:

- [Image 1: pytorch_2.1.0-cuda_12.1-py_3.10.6-ubuntu_22.04-x86_64](#)
- [Image 2: pytorch1.8-cuda10.2-cudnn7-ubuntu18.04](#)
- [Image 3: pytorch1.10-cuda10.2-cudnn7-ubuntu18.04](#)
- [Image 4: pytorch1.4-cuda10.1-cudnn7-ubuntu18.04](#)

Image 1: pytorch_2.1.0-cuda_12.1-py_3.10.6-ubuntu_22.04-x86_64

Table 9-21 Information about the image

AI Engine	Whether to Use GPUs (CUDA Version)	URL	Dependency	
PyTorch 2.1	Yes (CUDA 10.2)	swr.{region_id}.myhuaweicloud.com/atelier/pytorch_2_1:pytorch_2.1.0-cuda_12.1-py_3.10.6-ubuntu_22.04-x86_64-20250305173557-cb53968	PyPI package	Ubuntu package

AI Engine	Whether to Use GPUs (CUDA Version)	URL	Dependency	
			torch 2.1.0+cu121 torchvision 0.16.0+cu121 torchaudio 2.1.0+cu121 ipykernel 6.7.0 ipython 8.33.0 jupyter-client 7.4.9 ma-cli 1.2.3 matplotlib 3.7.3 moxing-framework 2.2.10 numpy 1.24.2 opencv-python 4.8.0.76 pandas 2.2.2 Pillow 10.4.0 pip 22.0.4 psutil 6.0.0 PyYAML 6.0.2 scipy 1.12.0 scikit-learn 1.5.1 tornado 6.4.2 tensorboard 2.18.0	automake build-essential ca-certificates cmake cpp curl ffmpeg g++ gcc gfortran git git-lfs grep libcudnn7 libcudnn7-dev libjpeg-dev:amd64 libjpeg8-dev:amd64 openssh-client openssh-server nginx pandoc python3 rpm screen tar tmux unzip vim wget zip

Image 2: pytorch1.8-cuda10.2-cudnn7-ubuntu18.04

Table 9-22 Information about the image

AI Engine	Whether to Use GPUs (CUDA Version)	URL	Dependency	
			PyPI package	Ubuntu package
PyTorch 1.8	Yes (CUDA 10.2)	swr.{region_id}.myhuaweicloud.com/atelier/pytorch_1_8:pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64-20220926104358-041ba2e		

AI Engine	Whether to Use GPUs (CUDA Version)	URL	Dependency	
			torch 1.8.0 torchvision 0.9.0 ipykernel 6.7.0 ipython 7.34.0 jupyter-client 7.4.4 ma-cli 1.2.3 matplotlib 3.5.1 modelarts 1.4.25 moxing-framework 2.1.0.5d9c87c8 numpy 1.19.5 opencv-python 4.1.2.30 pandas 1.1.5 Pillow 9.3.0 pip 21.0.1 psutil 5.8.0 PyYAML 5.1 scipy 1.5.2 scikit-learn 0.22.1 tornado 6.2 tensorboard 2.1.1	automake build-essential ca-certificates cmake cpp curl ffmpeg g++ gcc gfortran git git-lfs grep libcudnn7 libcudnn7-dev libjpeg-dev:amd64 libjpeg8-dev:amd64 openssh-client openssh-server nginx pandoc python3 rpm screen tar tmux unzip vim wget zip

Image 3: pytorch1.10-cuda10.2-cudnn7-ubuntu18.04

Table 9-23 Information about the image

AI Engine	Whether to Use GPUs (CUDA Version)	URL	Dependency	
			PyPI package	Ubuntu package
Pytorch 1.10	Yes (CUDA 10.2)	swr.{region_id}.myhuaweicloud.com/atelier/pytorch_1_10:pytorch_1.10.2-cuda_10.2-py_3.7-ubuntu_18.04-x86_64-20221008154718-2b3e39c		

AI Engine	Whether to Use GPUs (CUDA Version)	URL	Dependency	
			torch 1.10.2 torchvision 0.11.3 ipykernel 6.7.0 ipython 7.34.0 jupyter-client 7.4.4 ma-cli 1.2.3 matplotlib 3.5.1 modelarts 1.4.25 moxing-framework 2.1.0.5d9c87c8 numpy 1.19.5 opencv-python 4.1.2.30 pandas 1.1.5 Pillow 9.3.0 pip 21.0.1 psutil 5.8.0 PyYAML 5.1 scipy 1.5.2 scikit-learn 0.22.1 tornado 6.2	automake build-essential ca-certificates cmake cpp curl ffmpeg g++ gcc gfortran git git-lfs grep libcudnn7 libcudnn7-dev libjpeg-dev:amd64 libjpeg8-dev:amd64 openssh-client openssh-server nginx pandoc python3 rpm screen tar tmux unzip vim wget zip

Image 4: pytorch1.4-cuda10.1-cudnn7-ubuntu18.04

Table 9-24 Information about the image

AI Engine	Whether to Use GPUs (CUDA Version)	URL	Dependency	
			PyPI package	Ubuntu package
Pytorch 1.4	Yes (CUDA 10.1)	swr.{region_id}.myhuaweicloud.com/atelier/pytorch_1_4:pytorch_1.4-cuda_10.1-py37-ubuntu_18.04-x86_64-20220926104017-041ba2e		

AI Engine	Whether to Use GPUs (CUDA Version)	URL	Dependency	
			torch 1.4.0 torchvision 0.5.0 ipykernel 6.7.0 ipython 7.34.0 jupyter-client 7.4.7 ma-cli 1.2.3 matplotlib 3.5.1 modelarts 1.4.25 moxing-framework 2.1.0.5d9c87c8 numpy 1.19.5 opencv-python 4.1.2.30 pandas 1.1.5 Pillow 6.2.0 pip 21.0.1 psutil 5.8.0 PyYAML 5.1 scipy 1.5.2 scikit-learn 0.22.1 tornado 6.2 tensorboard 2.1.1	automake build-essential ca-certificates cmake cpp curl ffmpeg g++ gcc gfortran git git-lfs grep libcudnn7 libcudnn7-dev libjpeg-dev:amd64 libjpeg8-dev:amd64 openssh-client openssh-server nginx pandoc python3 rpm screen tar tmux unzip vim wget zip

x86-powered TensorFlow Base Images

TensorFlow contains two types of images:

- [Image 1: tensorflow2.1-cuda10.1-cudnn7-ubuntu18.04](#)
- [Image 2: tensorflow1.13-cuda10.0-cudnn7-ubuntu18.04](#)

Image 1: tensorflow2.1-cuda10.1-cudnn7-ubuntu18.04

Table 9-25 Information about the image

AI Engine	Whether to Use GPUs (CUDA Version)	URL	Dependency	
			PyPI package	Ubuntu package
Tensorflow 2.1	Yes (CUDA 10.1)	swr.{region_id}.myhuaweicloud.com/atelier/tensorflow_2_1:tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64-20220926144607-041ba2e		

AI Engine	Whether to Use GPUs (CUDA Version)	URL	Dependency	
			tensorflow 2.1.0 ipykernel 6.7.0 ipython 7.34.0 jupyter-client 7.4.4 ma-cli 1.2.3 matplotlib 3.5.1 modelarts 1.4.25 moxing-framework 2.1.0.5d9c87c8 numpy 1.19.5 opencv-python 4.1.2.30 pandas 1.1.5 Pillow 9.3.0 pip 21.0.1 psutil 5.8.0 PyYAML 5.1 scipy 1.5.2 scikit-learn 0.22.1 tornado 6.2 tensorboard 2.1.1	automake build-essential ca-certificates cmake cpp curl ffmpeg g++ gcc gfortran git git-lfs grep libcudnn7 libcudnn7-dev libjpeg-dev:amd64 libjpeg8-dev:amd64 openssh-client openssh-server nginx python3 rpm screen tar tmux unzip vim wget zip

Image 2: tensorflow1.13-cuda10.0-cudnn7-ubuntu18.04

Table 9-26 Information about the image

AI Engine	Whether to Use GPUs (CUDA Version)	URL	Dependency	
			PyPI package	Ubuntu package
Tensorflow 1.13-gpu	Yes (CUDA 10.0)	swr.{region_id}.myhuaweicloud.com/atelier/tensorflow_1_13:tensorflow_1.13-cuda_10.0-py_3.7-ubuntu_18.04-x86_64-20220926104358-041ba2e		

AI Engine	Whether to Use GPUs (CUDA Version)	URL	Dependency	
			tensorflow-gpu 1.13.1 ipykernel 6.7.0 ipython 7.34.0 jupyter-client 7.4.6 ma-cli 1.2.3 matplotlib 3.5.2 modelarts 1.4.25 moxing-framework 2.0.1.rc0.ff1c0c8 numpy 1.17.0 opencv-python 4.1.2.30 pandas 1.1.5 Pillow 6.2.0 pip 21.0.1 psutil 5.8.0 PyYAML 5.1 scipy 1.2.2 scikit-learn 0.22.1 tornado 6.2	automake build-essential ca-certificates cmake cpp curl ffmpeg g++ gcc gfortran git git-lfs grep libcudnn7 libcudnn7-dev libjpeg-dev:amd64 libjpeg8-dev:amd64 openssh-client openssh-server nginx python3 rpm screen tar tmux unzip vim wget zip

x86-powered MindSpore Base Images

MindSpore contains four types of images:

- [Image 1: mindspore1.7.0-cuda10.1-py3.7-ubuntu18.04](#)
- [Image 2: mindspore1.7.0-py3.7-ubuntu18.04](#)
- [Image 3: mindspore1.2.0-cuda10.1-cudnn7-ubuntu18.04](#)
- [Image 4: mindspore1.2.0-openmpi2.1.1-ubuntu18.04](#)

Image 1: mindspore1.7.0-cuda10.1-py3.7-ubuntu18.04

Table 9-27 Information about the image

AI Engine	Whether to Use GPUs (CUDA Version)	URL	Dependency	
			PyPI package	Ubuntu package
MindSpore-gpu 1.7.0	Yes (CUDA 10.1)	swr.{region_id}.myhuaweicloud.com/atelier/mindspore_1_7_0:mindspore_1.7.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64-20220926104017-041ba2e		

AI Engine	Whether to Use GPUs (CUDA Version)	URL	Dependency	
			mindspore-gpu 1.7.0 ipykernel 6.7.0 ipython 7.34.0 jupyter-client 7.4.4 ma-cli 1.2.3 matplotlib 3.5.1 modelarts 1.4.25 moxing-framework 2.1.0.5d9c87c8 numpy 1.19.5 pandas 1.1.5 Pillow 9.3.0 pip 21.0.1 psutil 5.8.0 PyYAML 5.1 scipy 1.5.2 scikit-learn 0.22.1 tornado 6.2 mindinsight 1.7.0 mindvision 0.1.0	automake build-essential ca-certificates cmake cpp curl ffmpeg g++ gcc gfortran git git-lfs grep libcudnn7 libcudnn7-dev libjpeg-dev:amd64 libjpeg8-dev:amd64 openssh-client openssh-server nginx python3 rpm screen tar tmux unzip vim wget zip

Image 2: mindspore1.7.0-py3.7-ubuntu18.04

Table 9-28 Information about the image

AI Engine	Whether to Use GPUs (CUDA Version)	URL	Dependency	
			PyPI package	Ubuntu package
Mind Spore 1.7.0	No	swr.{region_id}.myhuaweicloud.com/atelier/mindspore_1_7_0:mindspore_1.7.0-cpu-py_3.7-ubuntu_18.04-x86_64-20220926104017-041ba2e		

AI Engine	Whether to Use GPUs (CUDA Version)	URL	Dependency	
			mindspore 1.7.0 ipykernel 6.7.0 ipython 7.34.0 jupyter-client 7.4.6 ma-cli 1.2.3 matplotlib 3.5.1 modelarts 1.4.25 moxing-framework 2.1.0.5d9c87c8 numpy 1.19.5 pandas 1.1.5 Pillow 9.3.0 pip 21.0.1 psutil 5.8.0 PyYAML 5.1 scipy 1.5.2 scikit-learn 0.22.1 tornado 6.2 mindinsight 1.7.0 mindvision 0.1.0	automake build-essential ca-certificates cmake cpp curl ffmpeg g++ gcc gfortran git git-lfs grep libjpeg-dev:amd64 libjpeg8-dev:amd64 openssh-client openssh-server nginx python3 rpm screen tar tmux unzip vim wget zip

Image 3: mindspore1.2.0-cuda10.1-cudnn7-ubuntu18.04

Table 9-29 Information about the image

AI Engine	Whether to Use GPUs (CUDA Version)	URL	Dependency	
			PyPI package	Ubuntu package
Mind spore-gpu 1.2.0	Yes (CUDA 10.1)	swr.{region_id}.myhuaweicloud.com/atelier/mindspore_1_2_0:mindspore_1.2.0-py_3.7-cuda_10.1-ubuntu_18.04-x86_64-20220926104106-041ba2e		

AI Engine	Whether to Use GPUs (CUDA Version)	URL	Dependency	
			mindspore-gpu 1.2.0 ipykernel 6.7.0 ipython 7.34.0 jupyter-client 7.4.6 ma-cli 1.2.3 matplotlib 3.5.1 modelarts 1.4.25 moxing-framework 2.1.0.5d9c87c8 numpy 1.19.5 pandas 1.1.5 Pillow 6.2.0 pip 21.0.1 psutil 5.8.0 PyYAML 5.1 scipy 1.5.2 scikit-learn 0.22.1 tornado 6.2 mindinsight 1.2.0	automake build-essential ca-certificates cmake cpp curl ffmpeg g++ gcc gfortran git git-lfs grep libcudnn7 libcudnn7-dev libjpeg-dev:amd64 libjpeg8-dev:amd64 openssh-client openssh-server nginx python3 rpm screen tar tmux unzip vim wget zip

Image 4: mindspore1.2.0-openmpi2.1.1-ubuntu18.04

Table 9-30 Information about the image

AI Engine	Whether to Use GPUs (CUDA Version)	URL	Dependency	
			PyPI package	Ubuntu package
Mind spore 1.2.0	No	swr.{region_id}.myhuaweicloud.com/atelier/mindspore_1_2_0:mindspore_1.2.0-py_3.7-ubuntu_18.04-x86_64-20220926104106-041ba2e		

AI Engine	Whether to Use GPUs (CUDA Version)	URL	Dependency	
			mindspore 1.2.0 ipykernel 6.7.0 ipython 7.34.0 jupyter-client 7.4.6 ma-cli 1.2.3 matplotlib 3.5.1 modelarts 1.4.25 moxing-framework 2.1.0.5d9c87c8 numpy 1.19.5 pandas 1.1.5 Pillow 6.2.0 pip 21.0.1 psutil 5.8.0 PyYAML 5.1 scipy 1.5.2 scikit-learn 0.22.1 tornado 6.2 mindinsight 1.2.0	automake build-essential ca-certificates cmake cpp curl ffmpeg g++ gcc gfortran git git-lfs grep libjpeg-dev:amd64 libjpeg8-dev:amd64 openssh-client openssh-server nginx python3 rpm screen tar tmux unzip vim wget zip

x86-powered Custom Dedicated Base Images

ModelArts provides the following notebook base images powered by custom images (x86): **conda3-cuda10.2-cudnn7-ubuntu18.04** and **conda3-ubuntu18.04**.

These images do not have AI engines or related software packages. The image size is only 2 GB to 5 GB. You can use these images as base images and install your desired engine and dependency packages, improving scalability. In addition, these images are preset with some configurations required for starting the development environment. You can use these images after installing required software packages, without the need for any adaptations.

Such images are the most basic ones and have no component installed. They are small enough to facilitate image customization. If you need to use the OBS SDK, use ModelArts SDK instead to copy files. For details, see [Transferring Files](#).

Image 1: conda3-cuda10.2-cudnn7-ubuntu18.04

Table 9-31 Information about the image

AI Engine	Whether to Use GPUs (CUDA Version)	URL	Dependency	
None	Yes (CUDA 10.2)	swr.{region_id}.myhuaweicloud.com/atelier/user_defined_base:cuda_10.2-ubuntu_18.04-x86_64-20221008154718-2b3e39c	PyPI package	Ubuntu package
			ipykernel 6.7.0 ipython 7.34.0 jupyter-client 7.4.9 ma-cli 1.2.3 matplotlib 3.5.2 modelarts 1.4.25 moxing-framework 2.1.6.879ab2f4 numpy 1.21.6 pandas 1.3.5 Pillow 9.5.0 pip 20.3.3 psutil 5.9.4 PyYAML 6.0 scipy 1.7.3 tornado 6.2	automake build-essential ca-certificates cmake cpp curl g++ gcc gfortran grep libcudnn7 libcudnn7-dev nginx python3 rpm tar unzip vim wget zip

Image 2: conda3-ubuntu18.04

Table 9-32 Information about the image

AI Engine	Whether to Use GPUs (CUDA Version)	URL	Dependency	
None	No	swr. {region_id}.myhuaweicloud.com/atelier/ user_defined_base:ubuntu_18.04-x86_64-20221008154718-2b3e39c Example: CN North-Beijing4 swr.cn-north-4.myhuaweicloud.com/atelier/ user_defined_base:ubuntu_18.04-x86_64-20221008154718-2b3e39c CN East-Shanghai1 swr.cn-east-3.myhuaweicloud.com/atelier/ user_defined_base:ubuntu_18.04-x86_64-20221008154718-2b3e39c CN South-Guangzhou swr.cn-south-1.myhuaweicloud.com/atelier/ user_defined_base:ubuntu_18.04-x86_64-20221008154718-2b3e39c	PyPI package	Ubuntu package
			ipykernel 6.7.0 ipython 7.34.0 jupyter-client 7.4.9 ma-cli 1.2.3 matplotlib 3.5.2 modelarts 1.4.25 moxing-framework 2.1.6.879ab2f4 numpy 1.21.6 pandas 1.3.5 Pillow 9.5.0 pip 20.3.3 psutil 5.9.4 PyYAML 6.0 scipy 1.7.3 tornado 6.2	automake build-essential ca-certificates cmake cpp curl g++ gcc gfortran grep nginx python3 rpm tar unzip vim wget zip

Arm-powered MindSpore Base Images

Arm-powered MindSpore contains three types of images:

- [Image 1: mindspore_1.10.0-cann_6.0.1-py_3.7-euler_2.8.3](#)
- [Image 2: mindspore_1.9.0-cann_6.0.0-py_3.7-euler_2.8.3](#)
- [Image 3: mindspore1.7.0-cann5.1.0-py3.7-euler2.8.3](#)

Image 1: mindspore_1.10.0-cann_6.0.1-py_3.7-euler_2.8.3

Table 9-33 Information about the image

AI Engine	URL	Dependency	
		PyPI package	YUM package
Mindspore-Ascend 1.10.0	{region_id}.myhuaweicloud.com/atelier/mindspore_1_10_ascend:mindspore_1.10.0-cann_6.0.1-py_3.7-euler_2.8.3-aarch64-d910-20230303173945-815d627 Example: CN North-Beijing4 swr.cn-north-4.myhuaweicloud.com/atelier/mindspore_1_10_ascend:mindspore_1.10.0-cann_6.0.1-py_3.7-euler_2.8.3-aarch64-d910-20230303173945-815d627	mindspore-ascend 1.10.0 ipykernel 6.7.0 ipython 7.34.0 jupyter-client 7.4.5 ma-cli 1.2.3 matplotlib 3.5.1 modelarts 1.4.25 moxing-framework 2.0.1.rc0.ffd1c0c8 numpy 1.21.2 pandas 1.1.3 Pillow 9.4.0 pip 21.0.1 psutil 5.7.0 PyYAML 5.3.1 scipy 1.5.4 scikit-learn 0.24.0 tornado 6.2 mindinsight 1.9.0	cmake cpp curl ffmpeg g++ gcc git grep python3 rpm tar unzip wget zip

Image 2: mindspore_1.9.0-cann_6.0.0-py_3.7-euler_2.8.3

Table 9-34 Information about the image

AI Engine	URL	Dependency	
		PyPI package	YUM package
MindSpore 1.9.0	swr.{region_id}.myhuaweicloud.com/atelier/mindspore_1_9_ascend:mindspore_1.9.0-cann_6.0.0-py_3.7-euler_2.8.3-aarch64-d910-20221116111529	mindspore-ascend 1.9.0 ipykernel 6.7.0 ipython 7.34.0 jupyter-client 7.4.5 ma-cli 1.2.3 matplotlib 3.5.1 modelarts 1.4.25 moxing-framework 2.0.1.rc0.ff1c0c8 numpy 1.21.2 pandas 1.1.3 Pillow 9.3.0 pip 22.3.1 psutil 5.7.0 PyYAML 5.3.1 scipy 1.5.4 scikit-learn 0.24.0 tornado 6.2 mindinsight 1.9.0	cmake cpp curl ffmpeg g++ gcc git grep python3 rpm tar unzip wget zip

Image 3: mindspore1.7.0-cann5.1.0-py3.7-euler2.8.3

Table 9-35 Information about the image

AI Engine	URL	Dependency	
		PyPI package	YUM package
MindSpore-Ascend 1.7.0	swr.{region_id}.myhuaweicloud.com/atelier/mindspore_1_7_0:mindspore_1.7.0-cann_5.1.0-py_3.7-euler_2.8.3-aarch64-d910-20220906		

AI Engine	URL	Dependency	
		mindspore-ascend 1.7.0 ipykernel 5.3.4 ipython 7.34.0 jupyter-client 7.3.4 ma-cli 1.2.3 matplotlib 3.5.1 modelarts 1.4.25 moxing-framework 2.0.1.rc0.ffdf1c0c8 numpy 1.21.2 pandas 1.1.3 Pillow 9.2.0 pip 22.1.2 psutil 5.7.0 PyYAML 5.3.1 scipy 1.5.4 scikit-learn 0.24.0 tornado 6.2 mindinsight 1.7.0	cmake cpp curl ffmpeg g++ gcc git grep python3 rpm tar unzip wget zip

Arm-powered TensorFlow Base Images

Arm-powered TensorFlow contains two types of images:

- [Image 1: tensorflow1.15-mindspore1.7.0-cann5.1.0-euler2.8-aarch64](#)
- [Image 2: tensorflow1.15-cann5.1.0-py3.7-euler2.8.3](#)

Image 1: tensorflow1.15-mindspore1.7.0-cann5.1.0-euler2.8-aarch64

Table 9-36 Information about the image

AI Engine	URL	Dependency	
Mindspore-Ascend 1.7.0	swr. {region_id}.myhuaweicloud.com/ atelier/notebook2.0-mul-kernel- arm-ascend-cp37:5.0.1- c81-20220726	PyPI package	YUM package

AI Engine	URL	Dependency	
		mindspore-ascend 1.7.0 ipykernel 6.7.0 ipython 7.29.0 jupyter-client 7.0.6 ma-cli 1.2.3 matplotlib 3.1.2 modelarts 1.4.25 moxing-framework 2.0.0.rc2.4b57a67b numpy 1.17.5 pandas 1.1.3 Pillow 7.0.0 pip 21.2.4 psutil 5.7.0 PyYAML 5.3.1 scipy 1.5.4 scikit-learn 0.24.0 tornado 6.1 mindinsight 1.7.0	cmake cpp curl ffmpeg g++ gcc git grep python3 rpm tar unzip wget zip

Image 2: tensorflow1.15-cann5.1.0-py3.7-euler2.8.3

Table 9-37 Information about the image

AI Engine	Whether to Use Ascend (CANN Version)	URL	Dependency	
Tensorflow 1.15	Yes (CANN 5.1)	swr.{Region ID}.{Region domain name}.atelier/ tensorflow_1.15_ascend:tensorflow_1.15-cann_5.1.0-py_3.7-euler_2.8.3-aarch64-d910-20220906	PyPI package	YUM package

AI Engine	Whether to Use Ascend (CANN Version)	URL	Dependency	
			tensorflow 1.15.0 tensorboard 1.15.0 ipykernel 5.3.4 ipython 7.34.0 jupyter-client 7.3.4 ma-cli 1.2.3 matplotlib 3.5.1 modelarts 1.4.25 moxing-framework 2.0.1.rc0.ffd1c0c8 numpy 1.17.5 pandas 0.24.2 Pillow 9.2.0 pip 22.1.2 psutil 5.7.0 PyYAML 5.3.1 scipy 1.3.3 scikit-learn 0.20.0 tornado 6.2	ca-certificates. noarch cmake cpp curl gcc-c++ gcc gdb grep nginx python3 rpm tar unzip vim wget zip

9.2.4 Preset Dedicated Images for Training

ModelArts provides deep learning-powered base images such as TensorFlow, PyTorch, and MindSpore images. In these images, the software mandatory for running training jobs has been installed. If the software in the base images cannot meet your service requirements, create new images based on the base images and use the new images to create training jobs.

Available Training Base Images

The following table lists the preset training base images of ModelArts.

Table 9-38 ModelArts training base images

Engine	Version
PyTorch	pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64
TensorFlow	tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64
Horovod	horovod_0.20.0-tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64
	horovod_0.22.1-pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64
MPI	mindspore_1.3.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64
Ascend-Powered-Engine	tensorflow_1.15-cann_5.1.0-py_3.7-euler_2.8.3-aarch64
	mindspore_1.7.0-cann_5.1.0-py_3.7-euler_2.8.3-aarch64

 NOTE

Supported AI engines vary depending on regions.

PyTorch Base Images

This section describes preset PyTorch images.

Engine Version: [pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64](#)

Engine Version: [pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64](#)

- Image address: `swr.{Region}.myhuaweicloud.com/aip/pytorch_1_8:train-pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64-roma-20220309171256-40adcc1`
- Image creation time: 20220309171256 (yyyy-mm-dd-hh-mm-ss)
- Image system version: Ubuntu 18.04.4 LTS
- CUDA: 10.2.89
- cuDNN: 7.6.5.32
- Path and version of the Python interpreter: `/home/ma-user/anaconda3/envs/PyTorch-1.8/bin/python`, Python 3.7.10
- Third-party package installation path: `/home/ma-user/anaconda3/envs/PyTorch-1.8/lib/python3.7/site-packages`

TensorFlow Base Images

This section describes preset TensorFlow images.

- **Engine Version:** [tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64](#)

Engine Version: tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64

- Image address: `swr.{Region}.myhuaweicloud.com/aip/tensorflow_2_1:train-tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64-20210912152543-1e0838d`
- Image creation time: 20210912152543(yyyy-mm-dd-hh-mm-ss)
- Image system version: Ubuntu 18.04.4 LTS
- CUDA: 10.1.243
- cuDNN: 7.6.5.32
- Path and version of the Python interpreter: `/home/ma-user/anaconda3/envs/TensorFlow-2.1/bin/python`, Python 3.7.10
- Third-party package installation path: `/home/ma-user/anaconda3/envs/TensorFlow-2.1/lib/python3.7/site-packages`

Horovod Base Images

This section describes preset Horovod images.

- [Engine Version 1: horovod_0.20.0-tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64](#)
- [Engine Version 2: horovod_0.22.1-pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64](#)

Engine Version 1: horovod_0.20.0-tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64

- Image address: `swr.{Region}.myhuaweicloud.com/aip/horovod_tensorflow:train-horovod_0.20.0-tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64-20210912152543-1e0838d`
- Image creation time: 20210912152543 (yyyy-mm-dd-hh-mm-ss)
- Image system version: Ubuntu 18.04.4 LTS
- CUDA: 10.1.243
- cuDNN: 7.6.5.32
- Path and version of the Python interpreter: `/home/ma-user/anaconda3/envs/horovod_0.20.0-tensorflow_2.1.0/bin/python`, Python 3.7.10
- Third-party package installation path: `/home/ma-user/anaconda3/envs/horovod_0.20.0-tensorflow_2.1.0/lib/python3.7/site-packages`

Engine Version 2: horovod_0.22.1-pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64

- Image address: `swr.{Region}.myhuaweicloud.com/aip/horovod_pytorch:train-horovod_0.22.1-pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64-20210912152543-1e0838d`
- Image creation time: 20210912152543 (yyyy-mm-dd-hh-mm-ss)
- Image system version: Ubuntu 18.04.4 LTS
- CUDA: 11.1.1
- cuDNN: 8.0.5.39

- Path and version of the Python interpreter: **/home/ma-user/anaconda3/envs/horovod-0.22.1-pytorch-1.8.0/bin/python**, Python 3.7.10
- Third-party package installation path: **/home/ma-user/anaconda3/envs/horovod-0.22.1-pytorch-1.8.0/lib/python3.7/site-packages**

MPI Base Images

This section describes preset mindspore_1.3.0 images.

Engine Version: mindspore_1.3.0-cuda_10.1-py_3.7-ubuntu_1804-x86_64

Engine Version: mindspore_1.3.0-cuda_10.1-py_3.7-ubuntu_1804-x86_64

- Image address: swr.{*Region*}.myhuaweicloud.com/aip/mindspore_1_3_0:train-mindspore_1.3.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64-roma-20211104202338-f258e59
- Image creation time: 20211104202338 (yyyy-mm-dd-hh-mm-ss)
- Image system version: Ubuntu 18.04.4 LTS
- CUDA: 10.1.243
- cuDNN: 7.6.5.32
- Path and version of the Python interpreter: **/home/ma-user/anaconda3/envs/MindSpore-1.3.0-gpu/bin/python**, Python 3.7.10
- Third-party package installation path: **/home/ma-user/anaconda3/envs/MindSpore-1.3.0-gpu/lib/python3.7/site-packages**

9.2.5 Preset Dedicated Images for Inference

ModelArts inference provides a series of base images. You can create custom images based on these base images to deploy inference services.

x86 (CPU/GPU)-powered Base Images

Table 9-39 TensorFlow

AI Engine Version	Runtime Environment	Image	URI
2.1.0	CPU GPU(cuda10.1)	tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64	swr.{ <i>Region</i> ID}.myhuaweicloud.com/atelier/tensorflow_2_1:tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64-20221121111529-d65d817
1.15.5	CPU GPU(cuda11.4)	tensorflow_1.15.5-cuda_11.4-py_3.8-ubuntu_20.04-x86_64	swr.{ <i>Region</i> ID}.myhuaweicloud.com/aip/tensorflow_1_15:tensorflow_1.15.5-cuda_11.4-py_3.8-ubuntu_20.04-x86_64-20220524162601-50d6a18

AI Engine Version	Runtime Environment	Image	URI
2.6.0	CPU GPU(cuda11.2)	tensorflow_2.6.0-cuda_11.2-py_3.7-ubuntu_18.04-x86_64	swr.{ <i>Region</i> <i>ID</i> }.myhuaweicloud.com/aip/tensorflow_2_6:tensorflow_2.6.0-cuda_11.2-py_3.7-ubuntu_18.04-x86_64-20220524162601-50d6a18

Table 9-40 PyTorch

AI Engine Version	Runtime Environment	Image	URI
1.8.0	CPU GPU(cuda 10.2)	pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64	swr.{ <i>Region</i> <i>ID</i> }.myhuaweicloud.com/atelier/pytorch_1_8:pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64-20221118143845-d65d817
1.8.2	CPU GPU(cuda 11.1)	pytorch_1.8.2-cuda_11.1-py_3.7-ubuntu_18.04-x86_64	swr.{ <i>Region</i> <i>ID</i> }.myhuaweicloud.com/aip/pytorch_1_8:pytorch_1.8.2-cuda_11.1-py_3.7-ubuntu_18.04-x86_64-20220524162601-50d6a18

Table 9-41 MindSpore

AI Engine Version	Runtime Environment	Image name	URI
1.7.0	CPU	mindspore_1.7.0-cpu-py_3.7-ubuntu_18.04-x86_64	swr.{ <i>Region</i> <i>ID</i> }.myhuaweicloud.com/atelier/mindspore_1_7_0:mindspore_1.7.0-cpu-py_3.7-ubuntu_18.04-x86_64-20220702120711-8590b76

AI Engine Version	Runtime Environment	Image name	URI
1.7.0	GPU(cuda 10.1)	mindspore_1.7.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64	swr:{ <i>Region ID</i> }.myhuaweicloud.com/atelier/mindspore_1_7_0:mindspore_1.7.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64-20220702120711-8590b76
1.7.0	GPU(cuda 11.1)	mindspore_1.7.0-cuda_11.1-py_3.7-ubuntu_18.04-x86_64	swr:{ <i>Region ID</i> }.myhuaweicloud.com/atelier/mindspore_1_7_0:mindspore_1.7.0-cuda_11.1-py_3.7-ubuntu_18.04-x86_64-20220702120711-8590b76

TensorFlow (CPU/GPU)-powered Base Images

ModelArts provides the following inference base images powered by TensorFlow (CPU/GPU):

- **Engine Version 1:** [tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64](#)
- **Engine Version 2:** [tensorflow_1.15.5-cuda_11.4-py_3.8-ubuntu_20.04-x86_64](#)
- **Engine Version 3:** [tensorflow_2.6.0-cuda_11.2-py_3.7-ubuntu_18.04-x86_64](#)

Engine Version 1: tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64

- Image address: swr:{*Region ID*}.myhuaweicloud.com/atelier/tensorflow_2_1:tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64-20221121111529-d65d817
- Image creation time: 20220713110657 (yyyy-mm-dd-hh-mm-ss)
- Image system version: Ubuntu 18.04.4 LTS
- CUDA: 10.1.243
- cuDNN: 7.6.5.32
- Path and version of the Python interpreter: **/home/ma-user/anaconda3/envs/TensorFlow-2.1/bin/python**, Python 3.7.10
- Third-party package installation path: **/home/ma-user/anaconda3/envs/TensorFlow-2.1/lib/python3.7/site-packages**
- Certain pip installation packages:

Cython	0.29.21
easydict	1.9
Flask	2.0.1
grpcio	1.47.0

gunicorn	20.1.0
h5py	3.7.0
ipykernel	6.7.0
Jinja2	3.0.1
lxml	4.9.1
matplotlib	3.5.1
moxing-framework	2.1.0.5d9c87c8
numpy	1.19.5
opencv-python	4.1.2.30
pandas	1.1.5
Pillow	9.2.0
pip	22.1.2
protobuf	3.20.1
psutil	5.8.0
PyYAML	5.1
requests	2.27.1
scikit-learn	0.22.1
scipy	1.5.2
sklearn	0.0
tensorboard	2.1.1
tensorboardX	2.0
tensorflow	2.1.0
tensorflow-estimator	2.1.0
wheel	0.37.1
zipp	3.8.0
...	

- Certain apt installation packages:

```
apt
ca-certificates
cmake
cuda
curl
ethtool
fdisk
ffmpeg
g++
gcc
git
gpg
graphviz
libsm6
libxext6
libopencv-dev
libxrender-dev
libatlas3-base
libnuma-dev
libcap-dev
libssl-dev
liblz-dev
libbz2-dev
liblzma-dev
libboost-graph-dev
libsndfile1
libcurl4-openssl-dev
libopenblas-base
liblapack3
libopenblas-dev
libprotobuf-dev
libleveldb-dev
libsnapappy-dev
libhdf5-serial-dev
liblapacke-dev
libgflags-dev
libgoogle-glog-dev
liblmdb-dev
libatlas-base-dev
librdmacm1
libcap2-bin
libpq-dev
```

```
mysql-common
net-tools
nginx
openslide-tools
openssh-client
openssh-server
openssh-sftp-server
openssl
protobuf-compiler
redis-server
redis-tools
rpm
tar
tofrodos
unzip
vim
wget
zip
zlib1g-dev
...
```

Engine Version 2: tensorflow_1.15.5-cuda_11.4-py_3.8-ubuntu_20.04-x86_64

- Image address: swr.{*Region ID*}.myhuaweicloud.com/aip/tensorflow_1_15:tensorflow_1.15.5-cuda_11.4-py_3.8-ubuntu_20.04-x86_64-20220524162601-50d6a18
- Image creation time: 20220524162601 (yyyy-mm-dd-hh-mm-ss)
- Image system version: Ubuntu 20.04.4 LTS
- CUDA: 11.4.3
- cuDNN: 8.2.4.15
- Path and version of the Python interpreter: **/home/ma-user/anaconda3/envs/TensorFlow-1.15.5/bin/python**, Python 3.8.13
- Third-party package installation path: **/home/ma-user/anaconda3/envs/TensorFlow-1.15.5/lib/python3.8/site-packages**

- Certain pip installation packages:

```
Cython 0.29.21
psutil 5.9.0
matplotlib 3.5.1
protobuf 3.20.1
tensorflow 1.15.5+nv
Flask 2.0.1
grpcio 1.46.1
gunicorn 20.1.0
Pillow 9.0.1
tensorboard 1.15.0
PyYAML 6.0
pip 22.0.4
lxml 4.7.1
numpy 1.18.5
tensorflow-estimator 1.15.1
...
```

- Certain apt installation packages:

```
apt
ca-certificates
cmake
cuda
curl
ethtool
fdisk
ffmpeg
g++
gcc
```

```
git
gpg
graphviz
libsm6
libxext6
libopencv-dev
libxrender-dev
libatlas-base
libnuma-dev
libcap-dev
libssl-dev
liblz-dev
libbz2-dev
liblzma-dev
libboost-graph-dev
libsndfile1
libcurl4-openssl-dev
libopenblas-base
liblapack3
libopenblas-dev
libprotobuf-dev
libleveldb-dev
libsnapappy-dev
libhdf5-serial-dev
liblapacke-dev
libgflags-dev
libgoogle-glog-dev
liblmdb-dev
libatlas-base-dev
librdmacm1
libcap2-bin
libpq-dev
mysql-common
net-tools
nginx
openslide-tools
openssh-client
openssh-server
openssh-sftp-server
openssl
protobuf-compiler
redis-server
redis-tools
rpm
tar
tofrodos
unzip
vim
wget
zip
zlib1g-dev
...
```

Engine Version 3: tensorflow_2.6.0-cuda_11.2-py_3.7-ubuntu_18.04-x86_64

- Image address: `swr.{Region ID}.myhuaweicloud.com/aip/tensorflow_2_6:tensorflow_2.6.0-cuda_11.2-py_3.7-ubuntu_18.04-x86_64-20220524162601-50d6a18`
- Image creation time: 20220524162601 (yyyy-mm-dd-hh-mm-ss)
- Image system version: Ubuntu 18.04.4 LTS
- CUDA: 11.2.0
- cuDNN: 8.1.1.33
- Path and version of the Python interpreter: `/home/ma-user/anaconda3/envs/TensorFlow-2.6.0/bin/python`, Python 3.7.10

- Third-party package installation path: **/home/ma-user/anaconda3/envs/TensorFlow-2.6.0/lib/python3.7/site-packages**

- Certain pip installation packages:

```
Cython 0.29.21
requests 2.27.1
easydict 1.9
tensorboardX 2.0
tensorflow 2.6.0
Flask 2.0.1
grpcio 1.46.1
gunicorn 20.1.0
idna 3.3
tensorflow-estimator 2.9.0
pandas 1.1.5
Pillow 9.0.1
lxml 4.8.0
matplotlib 3.5.1
scikit-learn 0.22.1
psutil 5.8.0
PyYAML 5.1
numpy 1.17.0
opencv-python 4.1.2.30
protobuf 3.20.1
pip 21.2.2
...
```

- Certain apt installation packages:

```
apt
ca-certificates
cmake
cuda
curl
ethhtool
fdisk
ffmpeg
g++
gcc
git
gpg
graphviz
libsm6
libxext6
libopencv-dev
libxrender-dev
libatlas3-base
libnuma-dev
libcap-dev
libssl-dev
liblz-dev
libbz2-dev
liblzma-dev
libboost-graph-dev
libsndfile1
libcurl4-openssl-dev
libopenblas-base
liblapack3
libopenblas-dev
libprotobuf-dev
libleveldb-dev
libsnappy-dev
libhdf5-serial-dev
liblapack-dev
libgflags-dev
libgoogle-glog-dev
liblmdb-dev
libatlas-base-dev
librdmacm1
libcap2-bin
libpq-dev
```

```
mysql-common
net-tools
nginx
openslide-tools
openssh-client
openssh-server
openssh-sftp-server
openssl
protobuf-compiler
redis-server
redis-tools
rpm
tar
tofrodos
unzip
vim
wget
zip
zlib1g-dev
...
```

PyTorch (CPU/GPU)-powered Base Images

ModelArts provides the following inference base images powered by PyTorch (CPU/GPU):

- **Engine Version 1: [pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64](#)**
- **Engine Version 2: [pytorch_1.8.2-cuda_11.1-py_3.7-ubuntu_18.04-x86_64](#)**

Engine Version 1: [pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64](#)

- Image address: `swr.{Region ID}.myhuaweicloud.com/atelier/pytorch_1_8:pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64-20221118143845-d65d817`
- Image creation time: 20220713110657 (yyyy-mm-dd-hh-mm-ss)
- Image system version: Ubuntu 18.04.4 LTS
- CUDA: 10.2.89
- cuDNN: 7.6.5.32
- Path and version of the Python interpreter: **`/home/ma-user/anaconda3/envs/PyTorch-1.8/bin/python`**, Python 3.7.10
- Third-party package installation path: **`/home/ma-user/anaconda3/envs/PyTorch-1.8/lib/python3.7/site-packages`**
- Certain pip installation packages:

Cython	0.27.3
easydict	1.9
Flask	2.0.1
fonttools	4.34.4
unicorn	20.1.0
ipykernel	6.7.0
Jinja2	3.0.1
lxml	4.9.1
matplotlib	3.5.1
mmcv	1.2.7
moxing-framework	2.1.0.5d9c87c8
numpy	1.19.5
opencv-python	4.1.2.30
pandas	1.1.5
Pillow	9.2.0
pip	22.1.2
protobuf	3.20.1

psutil	5.8.0
PyYAML	5.1
requests	2.27.1
scikit-learn	0.22.1
scipy	1.5.2
sklearn	0.0
tensorboard	2.1.1
tensorboardX	2.0
torch	1.8.0
torchtext	0.5.0
torchvision	0.9.0
tornado	6.2
tqdm	4.64.0
traitlets	5.3.0
typing_extensions	4.3.0
urllib3	1.26.10
watchdog	2.0.0
wcwidth	0.2.5
Werkzeug	2.1.2
wheel	0.37.1
yapf	0.32.0
zipp	3.8.0
...	

- Certain apt installation packages:

```
apt
ca-certificates
cmake
cuda
curl
ethhtool
fdisk
ffmpeg
g++
gcc
git
gpg
graphviz
libsm6
libxext6
libopencv-dev
libxrender-dev
libatlas3-base
libnuma-dev
libcap-dev
libssl-dev
liblz-dev
libbz2-dev
liblzma-dev
libboost-graph-dev
libsndfile1
libcurl4-openssl-dev
libopenblas-base
liblapack3
libopenblas-dev
libprotobuf-dev
libleveldb-dev
libsnappy-dev
libhdf5-serial-dev
liblapacke-dev
libgflags-dev
libgoogle-glog-dev
liblmdb-dev
libatlas-base-dev
librdmacm1
libcap2-bin
libpq-dev
mysql-common
net-tools
nginx
```

```
openside-tools
openssh-client
openssh-server
openssh-sftp-server
openssl
protobuf-compiler
redis-server
redis-tools
rpm
tar
tofrodo
unzip
vim
wget
zip
zlib1g-dev
...
```

Engine Version 2: **pytorch_1.8.2-cuda_11.1-py_3.7-ubuntu_18.04-x86_64**

- Image address: `swr.{Region ID}.myhuaweicloud.com/aip/pytorch_1_8:pytorch_1.8.2-cuda_11.1-py_3.7-ubuntu_18.04-x86_64-20220524162601-50d6a18`
- Image creation time: 20220524162601 (yyyy-mm-dd-hh-mm-ss)
- Image system version: Ubuntu 18.04.4 LTS
- CUDA: 11.1.1
- cuDNN: 8.0.5.39
- Path and version of the Python interpreter: **/home/ma-user/anaconda3/envs/PyTorch-1.8.2/bin/python**, Python 3.7.10
- Third-party package installation path: **/home/ma-user/anaconda3/envs/PyTorch-1.8.2/lib/python3.7/site-packages**
- Certain pip installation packages:

```
Cython 0.27.3
mmcv 1.2.7
easydict 1.9
tensorboardX 2.0
torch 1.8.2+cu111
Flask 2.0.1
pandas 1.1.5
gunicorn 20.1.0
PyYAML 5.1
torchaudio 0.8.2
Pillow 9.0.1
psutil 5.8.0
lxml 4.8.0
matplotlib 3.5.1
torchvision 0.9.2+cu111
pip 21.2.2
protobuf 3.20.1
numpy 1.17.0
opencv-python 4.1.2.30
scikit-learn 0.22.1
...
```

- Certain apt installation packages:

```
apt
ca-certificates
cmake
cuda
curl
ethtool
fdisk
ffmpeg
```



```
g++  
gcc  
git  
gpg  
graphviz  
libsm6  
libxext6  
libopencv-dev  
libxrender-dev  
libatlas3-base  
libnuma-dev  
libcap-dev  
libssl-dev  
liblz-dev  
libbz2-dev  
liblzma-dev  
libboost-graph-dev  
libsndfile1  
libcurl4-openssl-dev  
libopenblas-base  
liblapack3  
libopenblas-dev  
libprotobuf-dev  
libleveldb-dev  
libsnappy-dev  
libhdf5-serial-dev  
liblapacke-dev  
libgflags-dev  
libgoogle-glog-dev  
liblmdb-dev  
libatlas-base-dev  
librdmacm1  
libcap2-bin  
libpq-dev  
mysql-common  
net-tools  
nginx  
openslide-tools  
openssh-client  
openssh-server  
openssh-sftp-server  
openssl  
protobuf-compiler  
redis-server  
redis-tools  
rpm  
tar  
tofrodo  
unzip  
vim  
wget  
zip  
zlib1g-dev  
...
```

MindSpore (CPU/GPU)-powered Base Images

ModelArts provides the following inference base images powered by MindSpore (CPU/GPU):

- [Engine Version 1: mindspore_1.7.0-cpu-py_3.7-ubuntu_18.04-x86_64](#)
- [Engine Version 2: mindspore_1.7.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64](#)
- [Engine Version 3: mindspore_1.7.0-cuda_11.1-py_3.7-ubuntu_18.04-x86_64](#)

Engine Version 1: mindspore_1.7.0-cpu-py_3.7-ubuntu_18.04-x86_64

- Image address: swr.{*Region ID*}.myhuaweicloud.com/atelier/mindspore_1_7_0:mindspore_1.7.0-cpu-py_3.7-ubuntu_18.04-x86_64-20220702120711-8590b76
- Image creation time: 20220702120711 (yyyy-mm-dd-hh-mm-ss)
- Image system version: Ubuntu 18.04.4 LTS
- Path and version of the Python interpreter: **/home/ma-user/anaconda3/envs/MindSpore/bin/python**, Python 3.7.10
- Third-party package installation path: **/home/ma-user/anaconda3/envs/MindSpore/lib/python3.7/site-packages**

- Certain pip installation packages:

```
cycler 0.11.0
easydict 1.9
Flask 2.0.1
grpcio 1.47.0
gunicorn 20.1.0
ipykernel 6.7.0
Jinja2 3.0.1
lxml 4.9.0
matplotlib 3.5.1
mindinsight 1.7.0
mindspore 1.7.0
mindvision 0.1.0
moxing-framework 2.1.0.5d9c87c8
numpy 1.17.0
opencv-contrib-python-headless 4.6.0.66
opencv-python-headless 4.6.0.66
pandas 1.1.5
Pillow 9.1.1
pip 22.1.2
protobuf 3.20.1
psutil 5.8.0
PyYAML 5.1
requests 2.27.1
scikit-learn 0.22.1
scipy 1.5.2
setuptools 62.6.0
sklearn 0.0
tensorboardX 2.0
threadpoolctl 3.1.0
tomli 2.0.1
tornado 6.1
tqdm 4.64.0
traitlets 5.3.0
treelib 1.6.1
urllib3 1.26.9
wheel 0.37.1
zipp 3.8.0
...
```

- Certain apt installation packages:

```
apt
ca-certificates
cmake
curl
ethtool
fdisk
ffmpeg
g++
gcc
git
gpg
graphviz
libsm6
```

```
libxext6
libopencv-dev
libxrender-dev
libatlas3-base
libnuma-dev
libcap-dev
libssl-dev
liblz-dev
libbz2-dev
liblzma-dev
libboost-graph-dev
libsndfile1
libcurl4-openssl-dev
libopenblas-base
liblapack3
libopenblas-dev
libprotobuf-dev
libleveldb-dev
libsnappy-dev
libhdf5-serial-dev
liblapacke-dev
libgflags-dev
libgoogle-glog-dev
liblmdb-dev
libatlas-base-dev
librdmacm1
libcap2-bin
libpq-dev
mysql-common
net-tools
nginx
openslide-tools
openssh-client
openssh-server
openssh-sftp-server
openssl
protobuf-compiler
redis-server
redis-tools
rpm
tar
tofrodos
unzip
vim
wget
zip
zlib1g-dev
...
```

Engine Version 2: mindspore_1.7.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64

- Image address: `swr.{Region ID}.myhuaweicloud.com/atelier/mindspore_1_7_0:mindspore_1.7.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64-20220702120711-8590b76`
- Image creation time: 20220702120711 (yyyy-mm-dd-hh-mm-ss)
- Image system version: Ubuntu 18.04.4 LTS
- CUDA: 10.1.243
- cuDNN: 7.6.5.32
- Path and version of the Python interpreter: **`/home/ma-user/anaconda3/envs/MindSpore/bin/python`**, Python 3.7.10
- Third-party package installation path: **`/home/ma-user/anaconda3/envs/MindSpore/lib/python3.7/site-packages`**

- Certain pip installation packages:

cycler	0.11.0
easydict	1.9
Flask	2.0.1
grpcio	1.47.0
gunicorn	20.1.0
ipykernel	6.7.0
Jinja2	3.0.1
lxml	4.9.0
matplotlib	3.5.1
mindinsight	1.7.0
mindspore	1.7.0
mindvision	0.1.0
moxing-framework	2.1.0.5d9c87c8
numpy	1.17.0
opencv-contrib-python-headless	4.6.0.66
opencv-python-headless	4.6.0.66
pandas	1.1.5
Pillow	9.1.1
pip	22.1.2
protobuf	3.20.1
psutil	5.8.0
PyYAML	5.1
requests	2.27.1
scikit-learn	0.22.1
scipy	1.5.2
setuptools	62.6.0
sklearn	0.0
tensorboardX	2.0
threadpoolctl	3.1.0
tomli	2.0.1
tornado	6.1
tqdm	4.64.0
traitlets	5.3.0
treelib	1.6.1
urllib3	1.26.9
wheel	0.37.1
zipp	3.8.0
...	

- Certain apt installation packages:

```

apt
ca-certificates
cmake
cuda
curl
ethtool
fdisk
ffmpeg
g++
gcc
git
gpg
graphviz
libsm6
libxext6
libopencv-dev
libxrender-dev
libatlas3-base
libnuma-dev
libcap-dev
libssl-dev
liblz-dev
libbz2-dev
liblzma-dev
libboost-graph-dev
libsndfile1
libcurl4-openssl-dev
libopenblas-base
liblapack3

```

```
libopenblas-dev
libprotobuf-dev
libleveldb-dev
libsnappy-dev
libhdf5-serial-dev
liblapacke-dev
libgflags-dev
libgoogle-glog-dev
liblmdb-dev
libatlas-base-dev
librdmacm1
libcap2-bin
libpq-dev
mysql-common
net-tools
nginx
openslide-tools
openssh-client
openssh-server
openssh-sftp-server
openssl
protobuf-compiler
redis-server
redis-tools
rpm
tar
tofrados
unzip
vim
wget
zip
zlib1g-dev
...
```

Engine Version 3: mindspore_1.7.0-cuda_11.1-py_3.7-ubuntu_18.04-x86_64

- Image address: swr.{*Region ID*}.myhuaweicloud.com/atelier/mindspore_1_7_0:mindspore_1.7.0-cuda_11.1-py_3.7-ubuntu_18.04-x86_64-20220702120711-8590b76
- Image creation time: 20220702120711 (yyyy-mm-dd-hh-mm-ss)
- Image system version: Ubuntu 18.04.4 LTS
- CUDA: 11.1.1
- cuDNN: 8.0.5.39
- Path and version of the Python interpreter: **/home/ma-user/anaconda3/envs/MindSpore/bin/python**, Python 3.7.10
- Third-party package installation path: **/home/ma-user/anaconda3/envs/MindSpore/lib/python3.7/site-packages**
- Certain pip installation packages:

```
cycler          0.11.0
easydict        1.9
Flask           2.0.1
grpcio          1.47.0
gunicorn        20.1.0
ipykernel       6.7.0
Jinja2          3.0.1
lxml            4.9.0
matplotlib      3.5.1
mindinsight     1.7.0
mindspore       1.7.0
mindvision      0.1.0
moxing-framework 2.1.0.5d9c87c8
numpy           1.17.0
opencv-contrib-python-headless 4.6.0.66
```

opencv-python-headless	4.6.0.66
pandas	1.1.5
Pillow	9.1.1
pip	22.1.2
protobuf	3.20.1
psutil	5.8.0
PyYAML	5.1
requests	2.27.1
scikit-learn	0.22.1
scipy	1.5.2
setuptools	62.6.0
sklearn	0.0
tensorboardX	2.0
threadpoolctl	3.1.0
tomli	2.0.1
tornado	6.1
tqdm	4.64.0
traitlets	5.3.0
treelib	1.6.1
urllib3	1.26.9
wheel	0.37.1
zipp	3.8.0
...	

- Certain apt installation packages:

apt
ca-certificates
cmake
cuda
curl
ethhtool
fdisk
ffmpeg
g++
gcc
git
gpg
graphviz
libsm6
libxext6
libopencv-dev
libxrender-dev
libatlas3-base
libnuma-dev
libcap-dev
libssl-dev
liblz-dev
libbz2-dev
liblzma-dev
libboost-graph-dev
libsndfile1
libcurl4-openssl-dev
libopenblas-base
liblapack3
libopenblas-dev
libprotobuf-dev
libleveldb-dev
libsnappy-dev
libhdf5-serial-dev
liblapacke-dev
libgflags-dev
libgoogle-glog-dev
liblmdb-dev
libatlas-base-dev
librdmacm1
libcap2-bin
libpq-dev
mysql-common
net-tools
nginx

```
openside-tools  
openssh-client  
openssh-server  
openssh-sftp-server  
openssl  
protobuf-compiler  
redis-server  
redis-tools  
rpm  
tar  
tofrodo  
unzip  
vim  
wget  
zip  
zlib1g-dev  
...
```

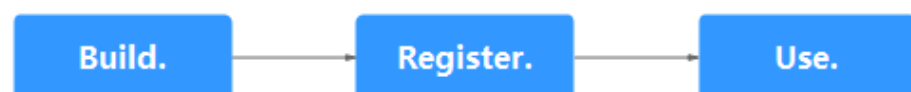
9.3 Creating a Custom Image for a Notebook Instance

9.3.1 Creating a Custom Image

Generally, you will need to reconstruct the ModelArts development environment, for example, by installing, upgrading, or uninstalling some packages. However, the root permission is required when certain packages are installed or upgraded. The running notebook instance does not have the root permission. As a result, you need to install the software that requires the root permission in the notebook instance, which is currently unavailable in the preset development environment. You can write a Dockerfile based on a preset base image or third-party image to customize your image.

Process for Creating a Custom Image

Figure 9-2 Creating a custom image (for scenarios 1 and 2)



Scenario 1: Build and register an image based on the preset image or third-party image in a notebook instance, configure the Docker environment on the server, and compile the Dockerfile. For details, see [Creating a Custom Image on ECS and Using It](#).

Scenario 2: Build and register a custom image for AI development based on the preset image or third-party image in a notebook instance and `ma-cli` commands. For details, see [ma-cli Image Building Command](#). In this case, the notebook instance is the platform for creating the image. For details, see [Creating a Custom Image Using Dockerfile](#).

Scenario 3: Create a notebook instance using a preset image, install custom software and dependencies on the preset image, and save the running instance

environment as a container image. For details, see [Creating a Custom Image Using the Image Saving Function](#).

Specifications for Custom Images

The base image for creating a custom image must meet either of the following conditions:

- It is an open-source image from the official website of Ascend or Docker Hub and it meets the following OS constraints:
x86: Ubuntu 18.04 or Ubuntu 20.04
Arm: Euler 2.8.3 or Euler 2.10.7

NOTE

There may be a compatibility issue for Ubuntu 20.04.6. Use an earlier version.

- If an image error occurs due to unmet requirements, check the image specifications and rectify the fault by referring to [Troubleshooting for Custom Images in Notebook Instances](#). If the fault persists, contact Huawei technical support.

Registering a Created Image

After a custom image is created, register it on the ModelArts **Image Management** page before using it in notebook.

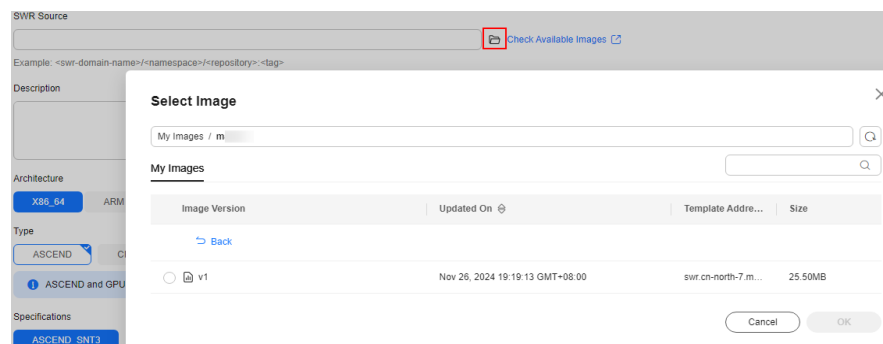
NOTE

Only the IAM users of the account can register and use the SWR images when the image type is **Private**.

Other users can register and use SWR images when the image type is **Public**.

- Log in to the ModelArts management console. In the navigation pane on the left, choose **Image Management**. Then, click **Register** on the displayed page.
- Configure parameters and click **Register**.
 - SWR Source:** Select a built image as the image source. You can copy the complete SWR address or click  to select the target image for registration.

Figure 9-3 Selecting an image source



- Architecture, Type, and Specifications:** Configure them based on the actual framework of the custom image.

3. View the registered image on the **Image Management** page.

9.3.2 Creating a Custom Image on ECS and Using It

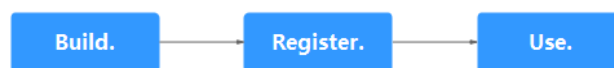
Application Scenarios and Process

You can write a Dockerfile based on a preset base image or third-party image to customize your image on ECS. Then, register the image to create a new development environment based on your needs.

This section describes how to install PyTorch 1.8, FFmpeg 3, and GCC 8 on an Ubuntu image to create a new AI development environment.

The following figure shows the whole process.

Figure 9-4 Creating and debugging an image



Specifications for Custom Images

The base image for creating a custom image must meet either of the following conditions:

- It is an open-source image from the official website of Ascend or Docker Hub and it meets the following OS constraints:
x86: Ubuntu 18.04 or Ubuntu 20.04
Arm: Euler 2.8.3 or Euler 2.10.7

NOTE

There may be a compatibility issue for Ubuntu 20.04.6. Use an earlier version.

- If an image error occurs due to unmet requirements, check the image specifications and rectify the fault by referring to [Troubleshooting for Custom Images in Notebook Instances](#). If the fault persists, contact Huawei technical support.

Procedure

1. Prepare a Linux environment. The following uses ECS as an example.
2. Create an image on ECS. The Dockerfile sample file is provided.
3. Upload the created image to SWR.
4. Register an SWR image on ModelArts.
5. Create a notebook instance and verify the new image.

Preparing a Docker Server and Configuring the Environment

Prepare a server with Docker enabled. If no such a server is available, create an ECS, buy an EIP, and install required software on it.

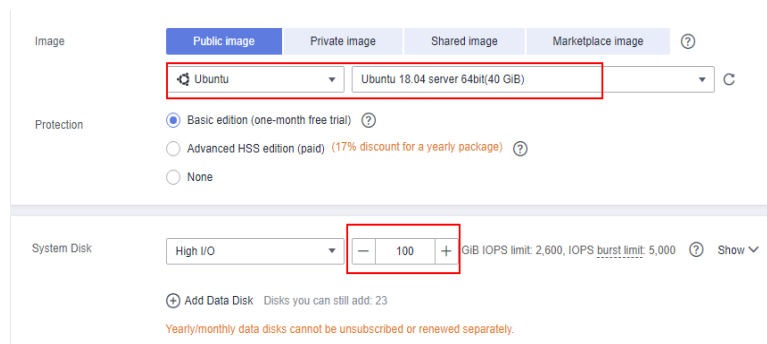
ModelArts provides Ubuntu scripts for you to install Docker easier.

NOTE

The operations on the local Linux server are the same as those on the ECS. For details, see this case.

1. Log in to the ECS console and click **Buy ECS**. Select a public image (an Ubuntu 18.04 image is recommended) and set the system disk to 100 GiB. For details, see [Purchasing and Logging In to a Linux ECS](#).

Figure 9-5 Selecting an image and a disk



2. Purchase an EIP and bind it to the ECS. For details, see [Configure Network](#).
3. Configure the VM environment.

- a. Run the following command on the Docker ECS to download the installation script:

```
wget https://cn-north-4-modelarts-sdk.obs.cn-north-4.myhuaweicloud.com/modelarts/custom-image-build/install_on_ubuntu1804.sh
```

NOTE

Only Ubuntu scripts are supported.

- b. Run the following command on the Docker ECS to configure the environment:

```
bash install_on_ubuntu1804.sh
```

Figure 9-6 Configured

```
Congratulations! The environment has been configured successfully.
```

```
source /etc/profile
```

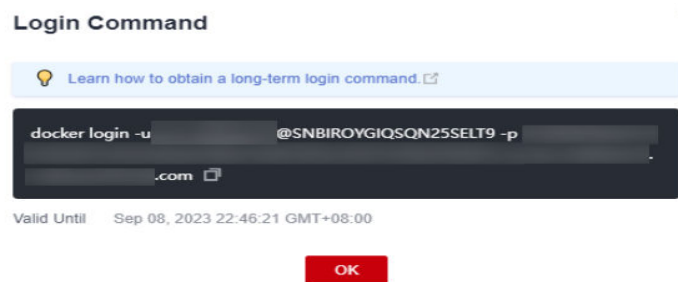
The installation script is executed to:

- i. Install Docker.
- ii. If the Docker ECS runs on GPUs, install nvidia-docker2 to mount the GPUs to the Docker container.

Creating a Custom Image

This section describes how to edit a Dockerfile, use it to create an image, and use the created image to create a notebook instance. For details about the Dockerfile, see [Dockerfile reference](#).

1. Querying Base Images (Skip This Step for Third-Party Images)
For details about ModelArts base images, see [Preset Dedicated Images in Notebook Instances](#). Check the image URL in the corresponding section based on the engine type of the preset image.
2. Access SWR.
 - a. Log in to the SWR console. In the navigation pane on the left, choose **Dashboard**, and click **Generate Login Command** in the upper right corner. On the displayed page, copy the login command.

Figure 9-7 Obtaining the login command**NOTE**

- The validity period of the generated login command is 24 hours. To obtain a long-term valid login command, see [Obtaining a Login Command with Long-Term Validity](#). After you obtain a long-term valid login command, your temporary login commands will still be valid as long as they are in their validity periods.
 - The domain name at the end of the login command is the image repository address. Record the address for later use.
- b. Run the login command on the machine where the container engine is installed. The message "Login Succeeded" will be displayed upon a successful login.
3. Pull a base image or third-party image. The following uses a third-party image as an example.

```
docker pull swr.ap-southeast-1.myhuaweicloud.com/notebook-xxx/ubuntu:18.04 #Your organization name and image
```
 4. Compile a Dockerfile.
Run the **vim** command to create a Dockerfile. If a ModelArts base image is used, see [Dockerfile on a ModelArts Base Image](#) for details about the Dockerfile.
If a third-party image is used, add user **ma-user** whose **UID** is **1000** and user group **ma-group** whose **GID** is **100**. For details, see [Dockerfile on a Non-ModelArts Base Image](#).
In this case, PyTorch 1.8, FFmpeg 3, and GCC 8 will be installed on an Ubuntu image to build an AI image.
 5. Build an image.
Run the **docker build** command to build a new image from the Dockerfile. The descriptions of the command parameters are as follows:
 - **-t** specifies the new image path, including region information, organization name, image name, and version. Set this parameter based

on the real-life scenario. Use a complete SWR address for debugging and registration.

- **-f** specifies the Dockerfile name. Set this parameter based on the real-life scenario.
- The period (.) at the end specifies that the context is the current directory. Set this parameter based on the real-life scenario.

```
docker build -t swr.ap-southeast-1.myhuaweicloud.com/notebook-xxx/pytorch_1_8:v1 -f Dockerfile .
```

Figure 9-8 Image created

```
Successfully built 495b3e4ff658
Successfully tagged swr.cn-north-4.myhuaweicloud.com/sdk-test/pytorch_1_8:v1
```

Registering a New Image

After an image is debugged, register it with ModelArts image management so that the image can be used in ModelArts.

1. Upload the image to SWR.

Log in to SWR first. For details, see [Logging in to SWR](#). Run the following command to push the image:

```
docker push swr.ap-southeast-1.myhuaweicloud.com/notebook-xxx/pytorch_1_8:v1
```

The image is then available on SWR.

Figure 9-9 Uploading the image to SWR

Tags					
Description Pull/Push Permissions Sharing Triggers Retention Synchronization Records					
Sync Delete					
Enter an image tag					
Tag	Size	Image Pull Command	Updated	Operation	
API-test-images	4.5 GB	docker pull	Sep 12, 2023 09:30:44 GMT+08:00	Scan	Sync More

2. Register an image.

Registering an image on the ModelArts console

Log in to the ModelArts console. In the navigation pane on the left, choose **Image Management** to access the image management page.

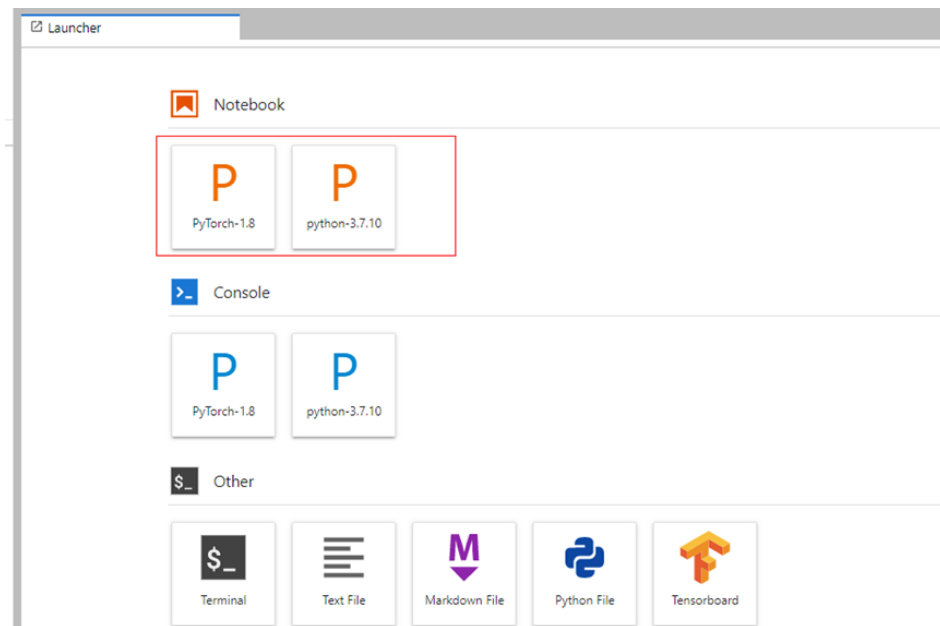
- a. Click **Register**. Set **SWR Source** to the image pushed to SWR in [step 1](#). Paste the complete SWR address or click  to select a private image from SWR for registration.
- b. Set **Architecture**, **Type**, and **Specifications** based on site requirements. The values must be those of the image source.

NOTE

When you register an image, ensure that the architecture and type are the same as those of the image source. Otherwise, the creation fails.

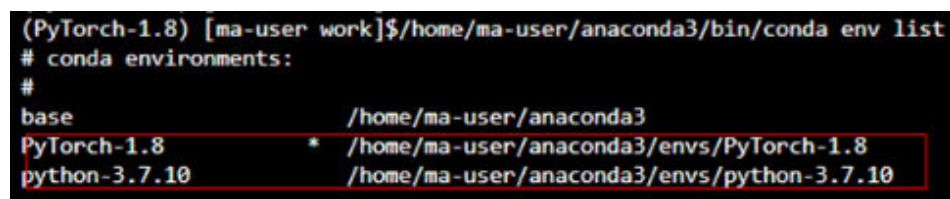
Using a New Image to Create a Development Environment

1. After the image is created, log in to the ModelArts console, go to the notebook tab, and choose the image registered in [2](#) to create a development environment.
2. Go to the notebook list, click **Open** to start the created development environment.

Figure 9-10 Accessing a development environment

3. Open a terminal to check the conda environment. For more information about conda, see the [official website](#).

Each kernel in the development environment is essentially a conda environment installed in `/home/ma-user/anaconda3/`. Run the `/home/ma-user/anaconda3/bin/conda env list` command to check the conda environment.

Figure 9-11 Checking the conda environment

Dockerfile on a ModelArts Base Image

Run the **vim** command to create a Dockerfile. If the base image is provided by ModelArts, the content of the Dockerfile is as follows:

```
FROM swr.ap-southeast-1.myhuaweicloud.com/atelier/notebook2.0-pytorch-1.4-kernel-cp37:3.3.3-release-v1-20220114

USER root
# section1: config apt source
RUN mv /etc/apt/sources.list /etc/apt/sources.list.bak && \
    echo -e "deb http://repo.huaweicloud.com/ubuntu/ bionic main restricted\ndeb http://\nrepo.huaweicloud.com/ubuntu/ bionic-updates main restricted\ndeb http://repo.huaweicloud.com/ubuntu/\nbionic universe\ndeb http://repo.huaweicloud.com/ubuntu/ bionic-updates universe\ndeb http://\nrepo.huaweicloud.com/ubuntu/ bionic multiverse\ndeb http://repo.huaweicloud.com/ubuntu/ bionic-updates\nmultiverse\ndeb http://repo.huaweicloud.com/ubuntu/ bionic-backports main restricted universe multiverse\n\ndeb http://repo.huaweicloud.com/ubuntu bionic-security main restricted\ndeb http://\nrepo.huaweicloud.com/ubuntu bionic-security universe\ndeb http://repo.huaweicloud.com/ubuntu bionic-\nsecurity multiverse" > /etc/apt/sources.list && \
    apt-get update
# section2: install ffmpeg and gcc
```

```
RUN apt-get -y install ffmpeg && \
    apt -y install gcc-8 g++-8 && \
    update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-8 80 --slave /usr/bin/g++ g++ /usr/bin/g++-8
&& \
    rm $HOME/.pip/pip.conf
USER ma-user
# section3: configure conda source and pip source
RUN echo -e "channels:\n - defaults\nshow_channel_urls: true\ndefault_channels:\n - https://
mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main\n - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/r
\n - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/msys2\ncustom_channels:\n conda-forge: https://
mirrors.tuna.tsinghua.edu.cn/anaconda/cloud\n msys2: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud
\n bioconda: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud\n menpo: https://
mirrors.tuna.tsinghua.edu.cn/anaconda/cloud\n pytorch: https://mirrors.tuna.tsinghua.edu.cn/anaconda/
cloud\n pytorch-lts: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud\n simpleitk: https://
mirrors.tuna.tsinghua.edu.cn/anaconda/cloud" > $HOME/.condarc && \
    echo -e "[global]\nindex-url = https://pypi.tuna.tsinghua.edu.cn/simple\n[install]\ntrusted-host = https://
pypi.tuna.tsinghua.edu.cn" > $HOME/.pip/pip.conf
# section4: create a conda environment(only support python=3.7) and install pytorch1.8
RUN source /home/ma-user/anaconda3/bin/activate && \
    conda create -y --name pytorch_1_8 python=3.7 && \
    conda activate pytorch_1_8 && \
    pip install torch==1.8.0 torchvision==0.9.0 torchaudio==0.8.0 && \
    conda deactivate
```

Dockerfile on a Non-ModelArts Base Image

If a third-party image is used, add user **ma-user** whose **UID** is **1000** and user group **ma-group** whose **GID** is **100** to the Dockerfile. If UID 1000 or GID 100 in the base image has been used by another user or user group, delete the user or user group. **The user and user group have been added to the Dockerfile in this case. You can directly use them.**

NOTE

You only need to set the user **ma-user** whose **UID** is **1000** and the user group **ma-group** whose **GID** is **100**, and grant the read, write, and execute permissions on the target directory to user **ma-user**.

Run the **vim** command to create a Dockerfile and add a third-party (non-ModelArts) image as the base image, for example, ubuntu 18.04. The content of the Dockerfile is as follows:

```
# Replace it with the actual image version.
FROM ubuntu:18.04
# Set the user ma-user whose UID is 1000 and the user group ma-group whose GID is 100
USER root
RUN default_user=$(getent passwd 1000 | awk -F ':' '{print $1}'); if [ ! -z $default_user ] && [ $default_user != "ma-user" ]; then \
    userdel -r $default_user; \
fi && \
if [ ! -z $default_group ] && [ $default_group != "ma-group" ]; then \
    groupdel -f $default_group; \
fi && \
groupadd -g 100 ma-group && useradd -d /home/ma-user -m -u 1000 -g 100 -s /bin/bash ma-user && \
# Grant the read, write, and execute permissions on the target directory to the user ma-user.
chmod -R 750 /home/ma-user

#Configure the APT source and install the ZIP and Wget tools (required for installing conda).
RUN mv /etc/apt/sources.list /etc/apt/sources.list.bak && \
    echo "deb http://repo.huaweicloud.com/ubuntu/ bionic main restricted\n deb http://
repo.huaweicloud.com/ubuntu/ bionic-updates main restricted\n deb http://repo.huaweicloud.com/ubuntu/
bionic universe\n deb http://repo.huaweicloud.com/ubuntu/ bionic-updates universe\n deb http://
repo.huaweicloud.com/ubuntu/ bionic multiverse\n deb http://repo.huaweicloud.com/ubuntu/ bionic-updates
multiverse\n deb http://repo.huaweicloud.com/ubuntu/ bionic-backports main restricted universe multiverse
\n deb http://repo.huaweicloud.com/ubuntu bionic-security main restricted\n deb http://
```

```
repo.huaweicloud.com/ubuntu bionic-security universe\ndeb http://repo.huaweicloud.com/ubuntu bionic-  
security multivers e" > /etc/apt/sources.list && \  
apt-get update && \  
apt-get install -y zip wget  
  
#Modifying the system Configuration of the image (required for creating the Conda environment)  
RUN rm /bin/sh && ln -s /bin/bash /bin/sh  
  
#Switch to user ma-user , download miniconda from the Tsinghua repository, and install miniconda in /  
home/ma-user.  
USER ma-user  
RUN cd /home/ma-user/ && \  
    wget --no-check-certificate https://mirrors.tuna.tsinghua.edu.cn/anaconda/miniconda/Miniconda3-4.6.14-  
Linux-x86_64.sh && \  
    bash Miniconda3-4.6.14-Linux-x86_64.sh -b -p /home/ma-user/anaconda3 && \  
    rm -rf Miniconda3-4.6.14-Linux-x86_64.sh  
  
#Configure the conda and pip sources  
RUN mkdir -p /home/ma-user/.pip && \  
    echo -e "channels:\n - defaults\nshow_channel_urls: true\ndefault_channels:\n - https://  
mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main\n - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/r  
\n - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/msys2" > /home/ma-user/.condarc && \  
    echo -e "[global]\nindex-url = https://pypi.tuna.tsinghua.edu.cn/simple\n[install]\ntrusted-host = https://  
pypi.tuna.tsinghua.edu.cn" > /home/ma-user/.pip/pip.conf  
  
#Create the conda environment and install the Python third-party package. The ipykernel package is  
mandatory for starting a kernel.  
RUN source /home/ma-user/anaconda3/bin/activate && \  
    conda create -y --name pytorch_1_8 python=3.7 && \  
    conda activate pytorch_1_8 && \  
    pip install torch==1.8.1 torchvision==0.9.1 && \  
    pip install ipykernel==6.7.0 && \  
    conda init bash && \  
    conda deactivate  
  
#Install FFmpeg and GCC  
USER root  
RUN apt-get -y install ffmpeg && \  
    apt -y install gcc-8 g++-8
```

9.3.3 Creating a Custom Image Using Dockerfile

Scenario

This example shows how to use **ma-cli** commands in ModelArts CLI to create and register a custom image for AI development with a preset PyTorch image. For details, see [ma-cli Image Building Command](#).

Procedure

1. Create a notebook instance.
2. Create a custom image in the notebook instance.
3. Register the image on ModelArts.
4. Create a notebook instance and verify the new image.

Creating a Notebook Instance

1. Log in to the ModelArts management console. In the navigation pane on the left, choose **Development Workspace > Notebook**. On the displayed page, click **Create Notebook**. Set **Image** to **Public image** and select a PyTorch

image. Retain the default values for other parameters. For details, see [Creating a Notebook Instance \(Default Page\)](#).

2. After the notebook instance is created and in the **Running** state, locate it in the notebook list, and click **Open** in the **Operation** column. On the displayed JupyterLab page, click **Terminal**.

Creating a Custom Image in a Notebook Instance

- Step 1** Configure authentication information, specify a profile, and enter the account information as prompted. For details, see [ma-cli Authentication](#).

```
ma-cli configure --auth PWD -P xxx
```

```
(MindSpore) [ma-user work]$ma-cli configure --auth PWD -P yuan
account []: hws
username []:
password:
```

- Step 2** Run `env|grep -i CURRENT_IMAGE_NAME` to query the image used by the current instance.

```
(PyTorch-1.8) [ma-user work]$env|grep -i CURRENT_IMAGE_NAME
CURRENT_IMAGE_NAME=swr.ap-southeast-1.myhuaweicloud.com/atelier/pytorch_1.8:pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64-20220926104358-041ba2e
```

- Step 3** Create an image.

1. Obtain the SWR address of the base image.

```
CURRENT_IMAGE_NAME=swr.ap-southeast-1.myhuaweicloud.com/atelier/
pytorch_1.8:pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-
x86_64-20220926104358-041ba2e
```

2. Load an image creation template.

Run the **ma-cli image get-template** command to query the image template.

```
(MindSpore) [ma-user work]$ma-cli image get-template
Template Name      Description
-----
customize_from_ubuntu_18.04_to_modelarts      Add ma-user, apt install packages and create a new conda environment with pip based on scratch ubuntu 18.04
upgrade_current_notebook_apt_packages          Install apt packages like ffmpeg, gcc-8, g++-8 based on current Notebook image
migrate_3rd_party_image_to_modelarts           General template for migrating your own or open source image to ModelArts
build_handwritten_number_inference_application Create a new AI application, used to generate an image to deploy and infer in ModelArts
update_dli_image_pip_package                   Install pip packages based on DLI image
forward_compat_cuda_11_image_to_modelarts      Migrate and forward compat cuda-11.x to ModelArts by upgrading only user-mode CUDA components
```

Run the **ma-cli image add-template** command to load the image template to the specified folder. The default path is where the current command is located. For example, load the **upgrade_current_notebook_apt_packages** image creation template.

```
ma-cli image add-template upgrade_current_notebook_apt_packages
```

```
(PyTorch-1.8) [ma-user work]$ma-cli image add-template upgrade_current_notebook_apt_packages
[ OK ] Successfully add configuration template [ upgrade_current_notebook_apt_packages ] under folder [ /home/ma-user/work/.ma/upgrade_current_notebook_apt_packages ]
```

3. Modify a Dockerfile.

The Dockerfile in this example is modified based on the base PyTorch image `pytorch1.8-cuda10.2-cudnn7-ubuntu18.04`, the image template `upgrade_current_notebook_apt_packages` is loaded, and GCC and G++ are upgraded.

After the image template is loaded, the Dockerfile will be automatically loaded in `.ma/upgrade_current_notebook_apt_packages`. Double-click the Dockerfile to open it. The content is as follows and you can modify it based on your needs:

```
FROM swr.ap-southeast-1.myhuaweicloud.com/atelier/pytorch_1.8:pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64-20220926104358-041ba2e
```



```
# Set proxy to download internet resources
ENV HTTP_PROXY=http://proxy.modelarts.com:80 \
  http_proxy=http://proxy.modelarts.com:80 \
  HTTPS_PROXY=http://proxy.modelarts.com:80 \
  https_proxy=http://proxy.modelarts.com:80

USER root

# Config apt source which can accelerate the apt package download speed. Also install ffmpeg and
gcc-8 in root mode
RUN cp -f /etc/apt/sources.list /etc/apt/sources.list.bak && \
  sed -i "s@http://.*archive.ubuntu.com@http://repo.huaweicloud.com@g" /etc/apt/sources.list && \
  sed -i "s@http://.*security.ubuntu.com@http://repo.huaweicloud.com@g" /etc/apt/sources.list && \
  apt update && \
  apt -y install ffmpeg && \
  apt install -y --no-install-recommends gcc-8 g++-8 && apt-get autoremove -y && \
  update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-8 80 --slave /usr/bin/g++ g++ /usr/bin/g+
+-8

# ModelArts requires ma-user as the default user to start image
USER ma-user
```

4. Build an image.

Run the **ma-cli image build** command to build an image with the Dockerfile. For details about the command, see [ma-cli Image Building Commands](#).

```
ma-cli image build .ma/upgrade_current_notebook_apt_packages/Dockerfile -swr notebook-test/
my_image:0.0.1 -P XXX
```

The Dockerfile is stored in **.ma/upgrade_current_notebook_apt_package/Dockerfile** and the new image is stored in **notebook-test/my_image:0.0.1** in SWR. **XXX** indicates the profile specified for authentication.

----End

Registering an Image

After an image is created, register it with ModelArts image management so that the image can be used in ModelArts.

Use either of the following methods:

- **Method 1:** Run the **ma-cli image register** command to register an image. Then, the information of the registered image is returned, including image ID and name, as shown in the following figure. For details about the command, see [ma-cli Image Building Commands](#).

```
ma-cli image register --swr-path=swr.ap-southeast-1.myhuaweicloud.com/notebook-test/
my_image:0.0.1 -P XXX
```

Figure 9-12 Registering an image

```
(MindSpore) [ma-user work]$ma-cli image register --swr-path=swr. -test/my_image:0.0.1 -P yf-
test
You are now in a notebook or devcontainer and cannot use 'ImageManagement.debug' to check your image. If you need to debug
it, please use a workstation.
[ OK ] Successfully registered this image and image information is
{
  "arch": "x86_64",
  "create_at": 1689046488158,
  "dev_services": [
    "NOTEBOOK",
    "SSH"
  ],
  "id": "lc5c9",
  "name": "my_image",
  "namespace": "yf-test",
  "origin": "CUSTOMIZE",
  "resource_categories": [
    "CPU",
    "GPU"
  ],
  "service_type": "UNKNOWN",
  "size": 3659922132,
  "status": "ACTIVE",
  "swr_path": "swr. test/my_image:0.0.1",
  "tag": "0.0.1",
  "tags": [],
  "type": "DEDICATED",
  "update_at": 1689046488158,
  "visibility": "PRIVATE",
  "workspace_id": "0"
}
```

- **Method 2:** Register the image on the ModelArts management console.
Log in to the ModelArts management console. In the navigation pane on the left, select **Image Management**. The **Image Management** page is displayed.
 - a. Click **Register**. Paste the complete SWR address or click  to select a private image from SWR for registration.
 - b. Set **Architecture** and **Type** based on the site requirements. The values must be those of the image source.

Creating and Using a Notebook Instance

After an image is registered, it is available for development environment creation. You can log in to the ModelArts management console, choose **DevEnviron** > **Notebook**, and select the image during creation.

9.3.4 Creating a Custom Image Using the Image Saving Function

To save a notebook environment image, do as follows: Create a notebook instance using a preset image, install custom software and dependencies on the base image, and save the running instance as a container image. After the image is saved, the default working directory is the `/` path in the root directory.

In the saved image, the installed dependencies are retained. The data stored in **home/ma-user/work** for persistent storage will not be stored. When you use VS Code for remote development, the plug-ins installed on the Server are retained.

 NOTE

- If the image fails to be saved, view the event on the notebook instance details page. For details, see [Viewing Notebook Events](#).
- The image to be saved should not be larger than 35 GB and there should be no more than 125 layers. Otherwise, the image may fail to be saved. For details, see [Space Allocation for Container Engines](#).
 - If a dedicated resource pool is used, log in to the ModelArts console. In the navigation pane on the left, choose **AI Dedicated Resource Pools > Elastic Clusters**. On the displayed page, configure the container engine size as needed. For details, see [Resizing a Dedicated Resource Pool](#).
 - If the fault persists, contact technical support.
- When you use the containerd image saving function, pay attention to the following:
 - Do not download files and save images at the same in an instance to prevent performance deterioration caused by resource competition.
 - If there is a file larger than 10 GB in an instance system folder (for example, `/opt/`), you should not use the image saving function, in case the image size is too large or the saving duration is too long.

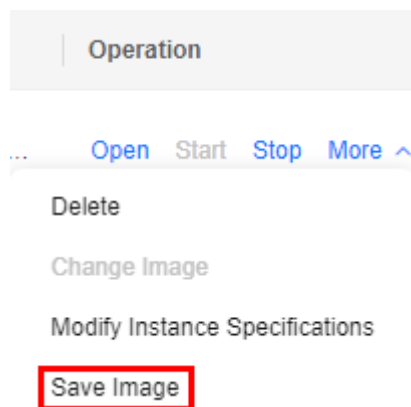
Prerequisites

The notebook instance is in **Running** state.

Saving an Image

1. Locate the target notebook instance in the list and choose **More > Save Image** in the **Operation** column.

Figure 9-13 Saving an image



2. In the displayed dialog box, configure the organization, image name, image version, and description, and click **OK**.

Choose an organization from the **Organization** drop-down list. If no organization is available, click **Create** to create one.

Users in an organization can share all images in the organization.
3. The image will be saved as a snapshot, and it will take about 5 minutes. During this period of time, do not perform any operations on the instance.

Figure 9-14 Saving as a snapshot**NOTICE**

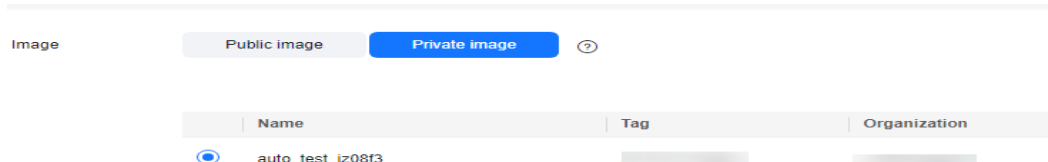
The time required for saving an image as a snapshot will be counted in the instance running duration. If the instance running duration expires before the snapshot is saved, saving the image will fail.

4. After the image is saved, the instance status changes to **Running**. You can view the image details on the image management page.
5. Click the name of the image to view its details.

Using a Custom Image to Create a Notebook Instance

The images saved from a notebook instance can be viewed on the **Image Management** page. You can use these images to create new notebook instances, which inherit the software configurations of the original notebook instances.

Method 1: On the **Create Notebook** page, click **Private Image** and select the saved image.

Figure 9-15 Selecting a custom image to create a notebook instance

Method 2: On the **Image Management** page, click the target image to access its details page. Then, click **Create Notebook**.

Which Data Can Be Saved When I Save an Image?

- Data that can be saved: Files and directories that are statically added to images during container building, for example, dependencies and the **/home/ma-user** directory are saved in the image environment.
- Data that cannot be saved: Mounting directories or data volumes that are dynamically connected to the host during container startup. You can run the **df -h** command to view the mounted dynamic directories. Data that is not in the **/** path will not be saved.

For example, data that is persistently stored in **home/ma-user/work** and data that is dynamically mounted to **/data** is not saved.

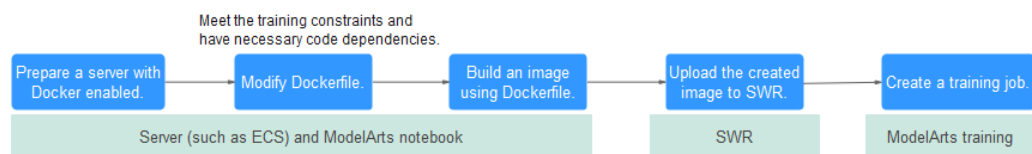
9.4 Creating a Custom Image for Model Training

9.4.1 Creating a Custom Training Image

If you have developed a model or training script locally but the AI engine you used is not supported by ModelArts, create a custom image and upload it to SWR. Then, use this image to create a training job on ModelArts and use the resources provided by ModelArts to train models.

Procedure

Figure 9-16 Creating a custom image for a training job



Scenario 1: If the preset images meet ModelArts training constraints but lack necessary code dependencies, install additional software packages.

For details, see [Creating a Custom Training Image Using a Preset Image](#).

Scenario 2: If the local images meet code dependency requirements but not ModelArts training constraints, adapt them to ModelArts.

For details, see [Migrating Existing Images to ModelArts](#).

Scenario 3: If neither the preset nor local images meet your needs, create an image that has necessary code dependencies and meet ModelArts constraints. For details, see the following cases:

[Creating a Custom Training Image \(PyTorch + CPU/GPU\)](#)

[Creating a Custom Training Image \(MPI + CPU/GPU\)](#)

[Creating a Custom Training Image \(Tensorflow + GPU\)](#)

Constraints on Custom Images of the Training Framework

- Use Ubuntu 18.04 for custom images to in case versions are not compatible.
- Do not use a custom image larger than 15 GB. The size should not exceed half of the container engine space of the resource pool. Otherwise, the start time of the training job is affected.

The container engine space of ModelArts public resource pool is 50 GB. By default, the container engine space of the dedicated resource pool is also 50 GB. You can customize the container engine space when creating a dedicated resource pool.

- The **uid** of the default user of a custom image must be **1000**.
- The GPU or Ascend driver cannot be installed in a custom image. When you select GPU resources to run training jobs, ModelArts automatically places the GPU driver in the **/usr/local/nvidia** directory in the training environment. When you select Ascend resources to run training jobs, ModelArts automatically places the Ascend driver in the **/usr/local/Ascend/driver** directory.

- x86- or Arm-based custom images can run only with specifications corresponding to their architecture.

Run the following command to check the CPU architecture of a custom image:

```
docker inspect {Custom image path} | grep Architecture
```

The following is the command output for an Arm-based custom image:

```
"Architecture": "arm64"
```

- If the name of a specification contains **Arm**, this specification is an Arm-based CPU architecture.
 - If the name of a specification does not contain **Arm**, this specification is an x86-based CPU architecture.
- The ModelArts backend does not support the download of open source installation packages. Install the dependency packages required for training in the custom image.
- Custom images can be used to train models in ModelArts only after they are uploaded to Software Repository for Container (SWR).

9.4.2 Creating a Custom Training Image Using a Preset Image

Principles

If you use a preset image to create a training job and you need to modify or add some software dependencies based on the preset image, you can create a custom image. In this case, on the training job creation page, select a preset image and choose **Customize** from the framework version drop-down list box.

The process of this method is the same as that of creating a training job based on a preset image. For example:

- The system automatically injects environment variables, as shown below:
 - `PATH=${MA_HOME}/anaconda/bin:${PATH}`
 - `LD_LIBRARY_PATH=${MA_HOME}/anaconda/lib:${LD_LIBRARY_PATH}`
 - `PYTHONPATH=${MA_JOB_DIR}:${PYTHONPATH}`
- The selected boot file will be automatically started using Python commands. Ensure that the Python environment is correct. The **PATH** environment variable is automatically injected. Run the following commands to check the Python version for the training job:
 - `export MA_HOME=/home/ma-user; docker run --rm {image} $ {MA_HOME}/anaconda/bin/python -V`
 - `docker run --rm {image} $(which python) -V`
- The system automatically adds hyperparameters associated with the preset image.

Creating a Training Image Using a Preset Image

ModelArts provides deep learning-powered base images such as TensorFlow, PyTorch, and MindSpore images. In these images, the software mandatory for running training jobs has been installed. If the software in the base images cannot meet your service requirements, create new images based on the base images and use the new images to create training jobs.

Perform the following operations to create an image using a training base image:

1. Install Docker. If the **docker images** command is executed, Docker has been installed. In this case, skip this step.

The following uses Linux x86_64 as an example to describe how to obtain the Docker installation package. Run the following command to install Docker:

```
curl -fsSL get.docker.com -o get-docker.sh
sh get-docker.sh
```

2. Create a folder named **context**.

```
mkdir -p context
```

3. Obtain the **pip.conf** file.

```
[global]
index-url = https://repo.huaweicloud.com/repository/pypi/simple
trusted-host = repo.huaweicloud.com
timeout = 120
```

4. Create an image based on a training base image provided by ModelArts. Save the edited Dockerfile in the **context** folder. For details about how to obtain a training base image, see [Preset Dedicated Images for Training](#).

FROM {Path to the training base image provided by ModelArts}

```
# Configure pip.
RUN mkdir -p /home/ma-user/.pip/
COPY --chown=ma-user:ma-group pip.conf /home/ma-user/.pip/pip.conf
```

```
# Configure the preset environment variables of the container image.
# Add the Python interpreter path to the PATH environment variable.
# Set PYTHONUNBUFFERED to 1 to prevent log loss.
ENV PATH=${ANACONDA_DIR}/envs/${ENV_NAME}/bin:$PATH \
    PYTHONUNBUFFERED=1
```

```
RUN /home/ma-user/anaconda/bin/pip install --no-cache-dir numpy
```

5. Create an image. Run the following command in the directory where the Dockerfile is stored to build the container image **training:v1**:

```
docker build . -t training:v1
```

6. Upload the new image to SWR.

- Log in to the SWR console and select the target region.
- Click **Create Organization** in the upper right corner and enter an organization name. In this case, **deep-learning** is used as an example. Replace it in subsequent commands with the actual organization name.
- Click **Generate Login Command** in the upper right corner to obtain a login command. Log in to ECS as user **root** and enter the login command.

Figure 9-17 Login command executed on ECS

```
root@ubuntu084-gpu-0:~# docker login -u 4942d7007f9106b3100e -p 6&&*****XXXXXXXXXXXXXXXXXXXXXXX sh.CI...
WARNING! Using --password via the CLI is insecure. Use --password-stored instead.
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store
```

- d. Log in to SWR and run the **docker tag** command to add tags to the image to be uploaded. In this case, **deep-learning** is used as an example. Replace it with the information configured in **a** for subsequent commands.

```
sudo docker tag tf-1.13.2:latest swr.Actual domain name.com/deep-learning/tf-1.13.2:latest
```

- e. Run the **docker push** command to upload the image.

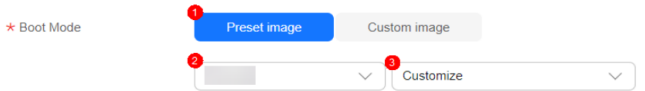
```
sudo docker push swr.Actual domain name.com/deep-learning/tf-1.13.2:latest
```

- f. After the image is uploaded, choose **My Images** in navigation pane on the left of the SWR console to view the uploaded custom images.

SWR URL of the custom image: `swr.<Region>.myhuaweicloud.com/deep-learning/tf-1.13.2:latest`

7. Create a training job on ModelArts.
 - a. Log in to the ModelArts console.
 - b. In the navigation pane on the left, choose **Model Training** > **Training Jobs**.
 - c. Click **Create Training Job**. On the displayed page, configure the parameters by referring to [Table 9-42](#). For details about the parameters, see [Creating a Production Training Job](#).

Table 9-42 Creating a training job

Parameter	Description
Algorithm Type	Mandatory. Select Custom algorithm .
Boot Mode	Mandatory. Select Preset image and choose the required framework and engine version. In this case, choose Customize for the engine version. 
Image	Select the image uploaded to SWR for container image.
Code Directory	Mandatory. Select the OBS directory where the training code file is stored. • Upload code to the OBS bucket beforehand. The total size of files in the directory cannot exceed 5 GB, the number of files cannot exceed 1,000, and the folder depth cannot exceed 32. • The training code file is automatically downloaded to the <code>\${MA_JOB_DIR}/demo-code</code> directory of the training container when the training job is started. demo-code is the last-level OBS directory for storing the code. For example, if Code Directory is set to <code>/test/code</code>, the training code file is downloaded to the <code>\${MA_JOB_DIR}/code</code> directory of the training container.
Boot File	Mandatory. Select the Python boot script of the training job in the code directory. ModelArts supports only the boot file written in Python. Therefore, the boot file must end with <code>.py</code>.

9.4.3 Migrating Existing Images to ModelArts

Scenario

An image is available on the local host and needs to be adapted on the cloud for ModelArts model training.

Procedure

1. Modify an existing image by referring to the following Dockerfile so that the image complies with specifications for custom images of the model training.
FROM {An existing image}

```
USER root
```

```
# If the user group whose GID is 100 already exists, delete the groupadd command.
```

```
RUN groupadd ma-group -g 100
```

```
# If the user whose UID is 1000 already exists, delete the useradd command.
```

```
RUN useradd -m -d /home/ma-user -s /bin/bash -g 100 -u 1000 ma-user
```

```
# Modify the permissions on image files so that user ma-user whose UID is 1000 can read and write the files.
```

```
RUN chown -R ma-user:100 {Path to the Python software package}
```

```
# Configure the preset environment variables of the container image.
```

```
# Set PYTHONUNBUFFERED to 1 to prevent log loss.
```

```
ENV PYTHONUNBUFFERED=1
```

```
# Configure the default user and working directory of the container image.
```

```
USER ma-user
```

```
WORKDIR /home/ma-user
```

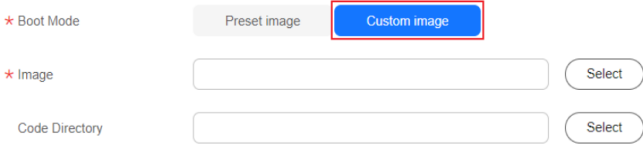
Note:

- a. Add the default user group **ma-group** (**gid = 100**) of the model training for the image.
 - If the user group whose **gid** is **100** already exists, the error message "groupadd: GID '100' already exists" may be displayed. You can run the **cat /etc/group | grep 100** command to check whether the user group whose GID is 100 exists.
 - If the user group whose **gid** is **100** already exists, skip this step and delete the command **RUN groupadd ma-group -g 100** from the Dockerfile.
- b. Add the default user **ma-user** (**uid = 1000**) of the model training for the image.
 - If the user whose **uid** is **1000** already exists, the error message "useradd: UID 1000 is not unique" may be displayed. You can run the **cat /etc/passwd | grep 1000** command to check whether the user whose UID is 1000 exists.
 - If the user whose **uid** is **1000** already exists, skip this step and delete the command **RUN useradd -d /home/ma-user -m -u 1000 -g 100 -s /bin/bash ma-user** from the Dockerfile.
- c. Modify the permissions on files in the image to allow **ma-user** whose **uid** is **1000** to read and write the files.

2. After editing the Dockerfile, run the following command to build an image:

```
docker build -f Dockerfile . -t {New image}
```
3. Upload the new image to SWR. For details, see [6](#).
4. Create a training job on ModelArts.
 - a. Log in to the ModelArts console.
 - b. In the navigation pane on the left, choose **Model Training** > **Training Jobs**.
 - c. Click **Create Training Job**. On the displayed page, configure the parameters by referring to [Table 9-42](#). For details about the parameters, see [Creating a Production Training Job](#).

Table 9-43 Creating a training job using a custom image

Parameter	Description
Algorithm Type	Mandatory. Select Custom algorithm .
Boot Mode	Mandatory. Select Custom image . 
Image	Mandatory. Select the image uploaded to SWR for container image.
Code Directory	OBS directory where the training code file is stored. Configure this parameter only if your custom image does not contain training code. • Upload code to the OBS bucket beforehand. The total size of files in the directory cannot exceed 5 GB, the number of files cannot exceed 1,000, and the folder depth cannot exceed 32. • The training code file is automatically downloaded to the \${MA_JOB_DIR}/demo-code directory of the training container when the training job is started. demo-code is the last-level OBS directory for storing the code. For example, if Code Directory is set to /test/code, the training code file is downloaded to the \${MA_JOB_DIR}/code directory of the training container.

Parameter	Description
User ID	User ID for running the container. The default value 1000 is recommended. If the UID needs to be specified, its value must be within the specified range. The UID ranges of different resource pools are as follows: - Public resource pool: 1000 to 65535 - Dedicated resource pool: 0 to 65535
Boot Command	Mandatory. Command for booting an image. When a training job is running, the boot command is automatically executed after the code directory is downloaded. - If the training boot script is a .py file, train.py for example, the boot command is as follows: <code>python \${MA_JOB_DIR}/demo-code/train.py</code> - If the training boot script is an .sh file, main.sh for example, the boot command is as follows: <code>bash \${MA_JOB_DIR}/demo-code/main.sh</code> You can use semicolons (;) and ampersands (&&) to combine multiple commands. demo-code in the command is the last-level OBS directory where the code is stored. Replace it with the actual one.
Local Code Directory	Specify the local directory of a training container. When a training starts, the system automatically downloads the code directory to this directory. (Optional) The default local code directory is /home/ma-user/modelarts/user-job-dir .
Work Directory	During training, the system automatically runs the cd command to execute the boot file in this directory.

9.4.4 Creating a Custom Training Image (PyTorch + Ascend)

This section describes how to create an image and use it for training on ModelArts. The AI engine used in the image is PyTorch, and the resources used for training are powered by Ascend in a dedicated resource pool.

Prerequisites

A Linux virtual or physical server with Docker 18.09.7 or later versions installed is available. The server can access the Internet and function as an image creation node.

Run the **docker pull**, **apt-get update/upgrade**, and **pip install** commands to check whether the node can access an external open-source software repository. If so, the node can access the Internet.

NOTICE

- The preceding servers must be Arm64-powered.
- It is a good practice to install Ubuntu 18.04 on the image creation node.
- In this section, the **/opt** directory is used for the image creation task. Ensure that the available storage of this directory is greater than 30 GB.
- For details about how to install Docker, see [Install Docker Engine on Ubuntu](#). Miniconda and TFLite installation packages are provided by third parties. ModelArts is not responsible for their security issues. If you have security requirements, harden the security of these packages, release them as files with the same names, and upload them to the image creation node.

Creating a Custom Image

Step 1 Check the Docker engine version.

```
docker version | grep -A 1 Engine
```

The command output is as follows:

```
Engine:  
Version: 18.09.0
```

NOTE

Use the Docker engine of the preceding version or later to create a custom image.

Step 2 Create a folder named **context**.

```
mkdir -p context
```

Step 3 Obtain the **pip.conf** file. In this example, the pip source provided by Huawei Mirrors is used, which is as follows:

```
[global]  
index-url =  
https://repo.huaweicloud.com/repository/pypi/simple  
trusted-host =  
repo.huaweicloud.com  
timeout = 120
```

Step 4 Obtain the APT source file **Ubuntu-Ports-bionic.list**. In this example, the APT source provided at Huawei Mirrors is used. Run the following command to obtain the APT source file:

```
wget -O Ubuntu-Ports-bionic.list --no-check-certificate  
https://repo.huaweicloud.com/repository/conf/Ubuntu-Ports-bionic.list
```

Step 5 Download the **Ascend-cann-nnae_7.0.0_linux-aarch64.run**, **torch-2.1.0-cp39-cp39-manylinux_2_17_aarch64.manylinux2014_aarch64.whl**, and **torch_npu-2.1.0.post7-cp39-cp39-manylinux_2_17_aarch64.manylinux2014_aarch64.whl** installation files.

- Download the **Ascend-cann-nnae_7.0.0_linux-aarch64.run** file. Click the following link based on your user type. Search for **CANN7** for version and click the **CANN7.0.0** link. On the displayed page, search for **Ascend-cann-nnae_7.0.0_linux-aarch64.run** and download it.

- For enterprise users, click [here](#) for download.
- For carrier users, click [here](#) for download.
- Click [here](#) to download the **torch-2.1.0-cp39-cp39-manylinux_2_17_aarch64.manylinux2014_aarch64.whl** file.
- Click [here](#) to download the **torch_npu-2.1.0.post7-cp39-cp39-manylinux_2_17_aarch64.manylinux2014_aarch64.whl** file.

 NOTE

ModelArts supports only the commercial CANN edition.

Step 6 Download the Miniconda3 installation file.

Click [here](#) to download the **Miniconda3-py39_24.5.0-0** installation file (for Python 3.9).

 NOTE

Download Python of a different version from [Miniconda3 File List](#). The MindSpore version must correspond to the Python version.

Step 7 Store the pip source file, **.list** file, **.run** file, **.whl** file, and Miniconda3 installation file in the **context** folder, which is as follows:

```
context
├── Ascend-cann-nnae_7.0.0_linux-aarch64.run
├── torch-2.1.0-cp39-cp39-manylinux_2_17_aarch64.manylinux2014_aarch64.whl
├── Miniconda3-py39_24.5.0-0-Linux-aarch64.sh
├── torch_npu-2.1.0.post7-cp39-cp39-manylinux_2_17_aarch64.manylinux2014_aarch64.whl
├── pip.conf
└── Ubuntu-Ports-bionic.list
```

Step 8 Write the Dockerfile of the container image.

Create an empty file named **Dockerfile** in the **context** folder and copy the following content to the file:

```
# The server on which the container image is created must access the Internet.
FROM ubuntu:18.04 AS builder

# The default user of the base container image is root.
# USER root

# Install OS dependencies.
COPY Ubuntu-Ports-bionic.list /tmp
RUN cp -a /etc/apt/sources.list /etc/apt/sources.list.bak && \
    mv /tmp/Ubuntu-Ports-bionic.list /etc/apt/sources.list && \
    echo > /etc/apt/apt.conf.d/00skip-verify-peer.conf "Acquire { https::Verify-Peer false }" && \
    apt-get update && \
    apt-get install -y \
    # utils
    ca-certificates vim curl \
    # CANN 7.0.0
    gcc g++ make cmake zlib1g zlib1g-dev openssl libsqlite3-dev libssl-dev libffi-dev unzip pciutils net-tools
    libblas-dev gfortran libblas3 libopenblas-dev \
    # MindSpore 2.2.0
    libgmp-dev && \
    apt-get clean && \
    mv /etc/apt/sources.list.bak /etc/apt/sources.list && \
    # Grant the write permission of the parent directory of the CANN 7.0.0 installation directory to ma-user.
    chmod o+w /usr/local

RUN useradd -m -d /home/ma-user -s /bin/bash -g 100 -u 1000 ma-user

# Configure the default user and working directory of the container image.
USER ma-user
```

```
WORKDIR /home/ma-user

# Use the PyPI configuration obtained at the open-source image website.
RUN mkdir -p /home/ma-user/.pip/
COPY --chown=ma-user:100 pip.conf /home/ma-user/.pip/pip.conf

# Copy the installation files to the /tmp directory in the base container image.
COPY --chown=ma-user:100 Miniconda3-py39_24.5.0-0-Linux-aarch64.sh /tmp

# https://conda.io/projects/conda/en/latest/user-guide/install/linux.html#installing-on-linux
# Install Miniconda3 in the /home/ma-user/miniconda3 directory of the base container image.
RUN bash /tmp/Miniconda3-py39_24.5.0-0-Linux-aarch64.sh -b -p /home/ma-user/miniconda3

ENV PATH=$PATH:/home/ma-user/miniconda3/bin

# Install the CANN 7.0.0 Python dependency package.
RUN pip install numpy~=1.19.2 decorator~=4.4.0 sympy~=1.5.1 cffi~=1.12.3 protobuf~=3.13.0 \
    attrs pyyaml pathlib2 scipy requests psutil absl-py

# Install CANN 7.0.0 in /usr/local/Ascend.
COPY --chown=ma-user:100 Ascend-cann-nnae_7.0.0_linux-aarch64.run /tmp
RUN chmod +x /tmp/Ascend-cann-nnae_7.0.0_linux-aarch64.run && \
    echo Y|/tmp/Ascend-cann-nnae_7.0.0_linux-aarch64.run --install --install-path=/usr/local/Ascend

# Install PyTorch 2.1.0.
COPY --chown=ma-user:100 torch-2.1.0-cp39-cp39-manylinux_2_17_aarch64.manylinux2014_aarch64.whl /tmp
RUN chmod +x /tmp/torch-2.1.0-cp39-cp39-manylinux_2_17_aarch64.manylinux2014_aarch64.whl && \
    pip install /tmp/torch-2.1.0-cp39-cp39-manylinux_2_17_aarch64.manylinux2014_aarch64.whl

# Install touch-npu.
COPY --chown=ma-user:100 torch_npu-2.1.0.post7-cp39-cp39-manylinux_2_17_aarch64.manylinux2014_aarch64.whl /tmp
RUN chmod +x /tmp/torch_npu-2.1.0.post7-cp39-cp39-manylinux_2_17_aarch64.manylinux2014_aarch64.whl && \
    pip install /tmp/torch_npu-2.1.0.post7-cp39-cp39-manylinux_2_17_aarch64.manylinux2014_aarch64.whl

# Create the container image.
FROM ubuntu:18.04

# Install OS dependencies.
COPY Ubuntu-Ports-bionic.list /tmp
RUN cp -a /etc/apt/sources.list /etc/apt/sources.list.bak && \
    mv /tmp/Ubuntu-Ports-bionic.list /etc/apt/sources.list && \
    echo > /etc/apt/apt.conf.d/00skip-verify-peer.conf "Acquire { https:Verify-Peer false }" && \
    apt-get update && \
    apt-get install -y \
    # utils
    ca-certificates vim curl \
    # CANN 7.0.RC1
    gcc g++ make cmake zlib1g zlib1g-dev openssl libsqlite3-dev libssl-dev libffi-dev unzip pciutils net-tools
    libblas-dev gfortran libblas3 libopenblas-dev \
    # MindSpore 2.2.0
    libgmp-dev && \
    apt-get clean && \
    mv /etc/apt/sources.list.bak /etc/apt/sources.list

RUN useradd -m -d /home/ma-user -s /bin/bash -g 100 -u 1000 ma-user

# Copy the directories from the builder stage to the directories with the same name in the current container image.
COPY --chown=ma-user:100 --from=builder /home/ma-user/miniconda3 /home/ma-user/miniconda3
COPY --chown=ma-user:100 --from=builder /home/ma-user/Ascend /home/ma-user/Ascend
COPY --chown=ma-user:100 --from=builder /home/ma-user/var /home/ma-user/var
COPY --chown=ma-user:100 --from=builder /usr/local/Ascend /usr/local/Ascend

# Configure the preset environment variables of the container image.
```

```
# Configure CANN environment variables.
# Configure Ascend driver environment variables.
# Set PYTHONUNBUFFERED to 1 to prevent log loss.
ENV PATH=$PATH:/usr/local/Ascend/nnae/latest/bin:/usr/local/Ascend/nnae/latest/compiler/cccec_compiler/
bin:/home/ma-user/miniconda3/bin \
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/Ascend/driver/lib64:/usr/local/Ascend/driver/lib64/
common:/usr/local/Ascend/driver/lib64/driver:/usr/local/Ascend/nnae/latest/lib64:/usr/local/Ascend/nnae/
latest/lib64/plugin/opskernel:/usr/local/Ascend/nnae/latest/lib64/plugin/nnengine \
PYTHONPATH=$PYTHONPATH:/usr/local/Ascend/nnae/latest/python/site-packages:/usr/local/Ascend/
nnae/latest/opp/op_impl/built-in/ai_core/tbe \
ASCEND_AICPU_PATH=$ASCEND_AICPU_PATH:/usr/local/Ascend/nnae/latest \
ASCEND_OPP_PATH=$ASCEND_OPP_PATH:/usr/local/Ascend/nnae/latest/opp \
ASCEND_HOME_PATH=$ASCEND_HOME_PATH:/usr/local/Ascend/nnae/latest \
PYTHONUNBUFFERED=1

# Configure the default user and working directory of the container image.
USER ma-user
WORKDIR /home/ma-user
```

For details about how to write a Dockerfile, see [official Docker documents](#).

Step 9 Verify that the Dockerfile has been created. The following shows the **context** folder:

```
context
├── Ascend-cann-nnae_7.0.0_linux-aarch64.run
├── Dockerfile
├── torch-2.1.0-cp39-cp39-manylinux_2_17_aarch64.manylinux2014_aarch64.whl
├── torch_npu-2.1.0.post7-cp39-cp39-manylinux_2_17_aarch64.manylinux2014_aarch64.whl
├── Miniconda3-py39_24.5.0-0-Linux-aarch64.sh
├── pip.conf
└── Ubuntu-Ports-bionic.list
```

Step 10 Create the container image. Run the following command in the directory where the Dockerfile is stored to create a container image:

```
docker build . -t pytorch:2.1.0-cann7.0.0
```

NOTE

If "connection refused" or "Client.Timeout exceeded" is reported when you access <https://registry-1.docker.io/v2/> during image creation, configure the Docker proxy.

```
vi /etc/docker/daemon.json
```

Add the following content to the file and save the file:

```
{
  "registry-mirrors":[
    "https://docker.m.daocloud.io",
    "https://docker.jianmuhub.com",
    "https://huecker.io",
    "https://dockerhub.timeweb.cloud",
    "https://dockerhub1.beget.com",
    "https://noohub.ru"]
}
```

Run the **systemctl daemon-reload** and **systemctl restart docker** commands in sequence.

Recreate

The following log shows that the image has been created.

```
Successfully tagged pytorch:2.1.0-cann7.0.0
```

----End

Uploading an Image to SWR

1. Log in to the SWR console and select a region. It must share the same region with ModelArts. Otherwise, the image cannot be selected.
2. Click **Create Organization** in the upper right corner and enter an organization name to create an organization. Customize the organization name. Replace the organization name **deep-learning** in subsequent commands with the actual organization name.
3. Click **Generate Login Command** in the upper right corner to obtain the login command. In this example, the temporary login command is copied.
4. Log in to the local environment as user **root** and enter the copied temporary login command.
5. Upload the image to SWR.
 - a. Run the **docker tag** command to add tags to the uploaded image:
Replace the region, domain, as well as organization name **deep-learning** with the actual values.

```
sudo docker tag pytorch:2.1.0-cnn7.0.0 swr.{region-id}-{domain}/deep-learning/pytorch:2.1.0-cnn7.0.0
```
 - b. Run the following command to upload the image:
Replace the region, domain, as well as organization name **deep-learning** with the actual values.

```
sudo docker push swr.{region-id}-{domain}/deep-learning/pytorch:2.1.0-cnn7.0.0
```
6. After the image is uploaded, choose **My Images** in navigation pane on the left of the SWR console to view the uploaded custom images.

Creating a Training Job on ModelArts

1. Log in to the ModelArts management console, check whether access authorization has been configured for your account. For details, see [Configuring Agency Authorization for ModelArts with One Click](#). If you have been authorized using access keys, clear the authorization and configure agency authorization.
2. In the navigation pane on the left, choose **Model Training > Training Jobs**. Then, click **Create Training Job**.
3. On the displayed page, configure parameters and click **Submit**.
 - **Algorithm Type:** Custom algorithm
 - **Boot Mode:** Custom image
 - **Image:** swr.cn-north-4.myhuaweicloud.com/deep-learning/pytorch:2.1.0-cnn7.0.0
 - **Code Directory:** directory where the boot script file is stored in OBS, for example, **obs://test-modelarts/pytorch/demo-code/**. The training code is automatically downloaded to the **`\${MA\_JOB\_DIR}/demo-code`** directory of the training container. **demo-code** (customizable) is the last-level directory of the OBS path.
 - **Boot Command:** **/home/ma-user/miniconda3/bin/python \$`\${MA\_JOB\_DIR}/demo-code/pytorch-verification.py`**. **demo-code** (customizable) is the last-level directory of the OBS path.
 - **Resource Pool:** Dedicated resource pool
 - **Resource Type:** Ascend with the required driver and firmware version

- **Job Log Path:** OBS path to stored training logs, for example, **obs://test-modelarts/pytorch/log/**.
4. Confirm the configurations of the training job and click **Submit**.
 5. Wait until the training job is created.

After you submit the job creation request, the system will automatically perform operations on the backend, such as downloading the container image and code directory and running the boot command. A training job requires a certain period of time for running. The duration ranges from dozens of minutes to several hours, depending on the service logic and selected resources. You can view log information on the job details page.

9.4.5 Creating a Custom Training Image (PyTorch + CPU/GPU)

This section describes how to create an image and use the image for training on the ModelArts platform. The AI engine used for training is PyTorch, and the resources are CPUs or GPUs.

NOTE

This section applies only to training jobs of the new version.

Scenario

In this example, write a Dockerfile to create a custom image on a Linux x86_64 server running Ubuntu 18.04.

Objective: Build and install container images of the following software and use the images and CPUs/GPUs for training jobs on ModelArts.

- Ubuntu 18.04
- CUDA 11.1
- Python 3.7.13
- pytorch-1.8.1

Procedure

Before using a custom image to create a training job, get familiar with Docker and have development experience. The following is the detailed procedure:

1. [Prerequisites](#)
2. [Step 1 Creating an OBS Bucket and Folder](#)
3. [Step 2 Preparing the Training Script and Uploading It to OBS](#)
4. [Step 3 Preparing a Host](#)
5. [Step 4 Creating a Custom Image](#)
6. [Step 5 Uploading an Image to SWR](#)
7. [Step 6 Creating a Training Job on ModelArts](#)

Prerequisites

You have registered a Huawei ID and enabled Huawei Cloud services, and the account is not in arrears or frozen.

Step 1 Creating an OBS Bucket and Folder

Create a bucket and the folders in OBS for storing the sample dataset and training code. [Table 9-44](#) lists the folders to be created. Replace the bucket name and folder names in the example with actual names.

For details about how to create an OBS bucket and folder, see [Creating a Bucket](#) and [Creating a Folder](#).

Ensure that the OBS directory you use and ModelArts are in the same region.

Table 9-44 Required OBS folders

Folder	Usage
obs://test-modelarts/pytorch/demo-code/	Stores the training script.
obs://test-modelarts/pytorch/log/	Stores training log files.

Step 2 Preparing the Training Script and Uploading It to OBS

Prepare the training script **pytorch-verification.py** and upload it to the **obs://test-modelarts/pytorch/demo-code/** folder of the OBS bucket.

The **pytorch-verification.py** file contains the following information:

```
import torch
import torch.nn as nn

x = torch.randn(5, 3)
print(x)

available_dev = torch.device("cuda") if torch.cuda.is_available() else torch.device("cpu")
y = torch.randn(5, 3).to(available_dev)
print(y)
```

Step 3 Preparing a Host

Obtain a Linux x86_64 server running Ubuntu 18.04. Either an ECS or your local PC will do.

For details about how to purchase an ECS, see [Purchasing and Logging In to a Linux ECS](#). Set **CPU Architecture** to **x86** and **Image** to **Public image**. Ubuntu 18.04 images are recommended.

Step 4 Creating a Custom Image

Create a container image with the following configurations and use the image to create a training job on ModelArts:

- Ubuntu 18.04
- CUDA 11.1
- Python 3.7.13
- pytorch-1.8.1

This section describes how to write a Dockerfile to create a custom image.

1. Install Docker.

The following uses the Linux x86_64 OS as an example to describe how to obtain the Docker installation package. For more details about how to install Docker, Run the following commands to install Docker:

```
curl -fsSL get.docker.com -o get-docker.sh
sh get-docker.sh
```

If the **docker images** command is executed, Docker has been installed. In this case, skip this step.

2. Run the following command to check the Docker Engine version:

```
docker version | grep -A 1 Engine
```

The following information is displayed:

```
...
Engine:
Version:      18.09.0
```

 NOTE

Use the Docker engine of the preceding version or later to create a custom image.

3. Create a folder named **context**.

```
mkdir -p context
```

4. Obtain the **pip.conf** file. In this example, the pip source provided by Huawei open-source image site is used. The content of **pip.conf** is as follows:

```
[global]
index-url = https://repo.huaweicloud.com/repository/pypi/simple
trusted-host = repo.huaweicloud.com
timeout = 120
```

 NOTE

To obtain **pip.conf**, go to Huawei Mirrors at <https://mirrors.huaweicloud.com/home> and search for **pypi**.

5. Download the **torch*.whl** files. Download the following .whl files from https://download.pytorch.org/whl/torch_stable.html:

- torch-1.8.1+cu111-cp37-cp37m-linux_x86_64.whl
- torchaudio-0.8.1-cp37-cp37m-linux_x86_64.whl
- torchvision-0.9.1+cu111-cp37-cp37m-linux_x86_64.whl

 NOTE

The URL code of the + symbol is %2B. When searching for a file in the above website, replace the + symbol in the file name with %2B.

For example, **torch-1.8.1%2Bcu111-cp37-cp37m-linux_x86_64.whl**.

6. Download the Miniconda3 py37 4.12.0 installation file (Python 3.7.13) from https://repo.anaconda.com/miniconda/Miniconda3-py37_4.12.0-Linux-x86_64.sh.

7. Store the pip source file, torch*.whl file, and Miniconda3 installation file in the **context** folder, which is as follows:

```
context
├── Miniconda3-py37_4.12.0-Linux-x86_64.sh
├── pip.conf
├── torch-1.8.1+cu111-cp37-cp37m-linux_x86_64.whl
├── torchaudio-0.8.1-cp37-cp37m-linux_x86_64.whl
└── torchvision-0.9.1+cu111-cp37-cp37m-linux_x86_64.whl
```

8. Write the container image Dockerfile.

Create an empty file named **Dockerfile** in the **context** folder and copy the following content to the file:

```
# The host must be connected to the public network for creating a container image.

# Base container image at https://github.com/NVIDIA/nvidia-docker/wiki/CUDA
#
# https://docs.docker.com/develop/develop-images/multistage-build/#use-multi-stage-builds
# require Docker Engine >= 17.05
#
# builder stage
FROM nvidia/cuda:11.1.1-runtime-ubuntu18.04 AS builder

# The default user of the base container image is root.
# USER root

# Use the PyPI configuration provided by the open-source image site.
RUN mkdir -p /root/.pip/
COPY pip.conf /root/.pip/pip.conf

# Copy the installation files to the /tmp directory in the base container image.
COPY Miniconda3-py37_4.12.0-Linux-x86_64.sh /tmp
COPY torch-1.8.1+cu111-cp37-cp37m-linux_x86_64.whl /tmp
COPY torchvision-0.9.1+cu111-cp37-cp37m-linux_x86_64.whl /tmp
COPY torchaudio-0.8.1-cp37-cp37m-linux_x86_64.whl /tmp

# https://conda.io/projects/conda/en/latest/user-guide/install/linux.html#installing-on-linux
# Install Miniconda3 to the /home/ma-user/miniconda3 directory of the base container image.
RUN bash /tmp/Miniconda3-py37_4.12.0-Linux-x86_64.sh -b -p /home/ma-user/miniconda3

# Install torch*.whl using the default Miniconda3 Python environment in /home/ma-user/miniconda3/bin/pip.
RUN cd /tmp && \
    /home/ma-user/miniconda3/bin/pip install --no-cache-dir \
    /tmp/torch-1.8.1+cu111-cp37-cp37m-linux_x86_64.whl \
    /tmp/torchvision-0.9.1+cu111-cp37-cp37m-linux_x86_64.whl \
    /tmp/torchaudio-0.8.1-cp37-cp37m-linux_x86_64.whl

# Create the final container image.
FROM nvidia/cuda:11.1.1-runtime-ubuntu18.04

# Install vim and cURL in the open-source image site.
RUN cp -a /etc/apt/sources.list /etc/apt/sources.list.bak && \
    sed -i "s@http://.*archive.ubuntu.com@http://repo.huaweicloud.com@" /etc/apt/sources.list && \
    sed -i "s@http://.*security.ubuntu.com@http://repo.huaweicloud.com@" /etc/apt/sources.list && \
    apt-get update && \
    apt-get install -y vim curl && \
    apt-get clean && \
    mv /etc/apt/sources.list.bak /etc/apt/sources.list

# Add user ma-user (UID = 1000, GID = 100).
# A user group whose GID is 100 of the base container image exists. User ma-user can directly use it.
RUN useradd -m -d /home/ma-user -s /bin/bash -g 100 -u 1000 ma-user

# Copy the /home/ma-user/miniconda3 directory from the builder stage to the directory with the same name in the current container image.
COPY --chown=ma-user:100 --from=builder /home/ma-user/miniconda3 /home/ma-user/miniconda3

# Configure the preset environment variables of the container image.
# Set PYTHONUNBUFFERED to 1 to avoid log loss.
ENV PATH=$PATH:/home/ma-user/miniconda3/bin \
    PYTHONUNBUFFERED=1

# Set the default user and working directory of the container image.
USER ma-user
WORKDIR /home/ma-user
```

For details about how to write a Dockerfile, see [official Docker documents](#).

9. Verify that the Dockerfile has been created. The following shows the **context** folder:

```
context
├── Dockerfile
├── Miniconda3-py37_4.12.0-Linux-x86_64.sh
├── pip.conf
├── torch-1.8.1+cu111-cp37-cp37m-linux_x86_64.whl
├── torchaudio-0.8.1-cp37-cp37m-linux_x86_64.whl
└── torchvision-0.9.1+cu111-cp37-cp37m-linux_x86_64.whl
```

10. Create the container image. Run the following command in the directory where the Dockerfile is stored to build the container image **pytorch:1.8.1-cuda11.1**:

```
docker build . -t pytorch:1.8.1-cuda11.1
```

The following log information displayed during image creation indicates that the image has been created.

```
Successfully tagged pytorch:1.8.1-cuda11.1
```

Step 5 Uploading an Image to SWR

1. Log in to the SWR console and select a region. It must share the same region with ModelArts. Otherwise, the image cannot be selected.
2. Click **Create Organization** in the upper right corner and enter an organization name to create an organization. Customize the organization name. Replace the organization name **deep-learning** in subsequent commands with the actual organization name.
3. Click **Generate Login Command** in the upper right corner to obtain the login command. In this example, the temporary login command is copied.
4. Log in to the local environment as user **root** and enter the copied temporary login command.
5. Upload the image to SWR.
 - a. Run the following command to tag the uploaded image:
#Replace the region and domain information with the actual values, and replace the organization name **deep-learning** with your custom value.

```
sudo docker tag pytorch:1.8.1-cuda11.1 swr.{region-id}:{domain}/deep-learning/pytorch:1.8.1-cuda11.1
```
 - b. Run the following command to upload the image:
#Replace the region and domain information with the actual values, and replace the organization name **deep-learning** with your custom value.

```
sudo docker push swr.{region-id}:{domain}/deep-learning/pytorch:1.8.1-cuda11.1
```
6. After the image is uploaded, choose **My Images** in navigation pane on the left of the SWR console to view the uploaded custom images.

Step 6 Creating a Training Job on ModelArts

1. Log in to the ModelArts management console and check whether access authorization has been configured for your account. For details, see [Configuring Agency Authorization for ModelArts with One Click](#). If you have been authorized using access keys, clear the authorization and configure agency authorization.
2. In the navigation pane on the left, choose **Model Training > Training Jobs**. The training job list is displayed by default.
3. On the **Create Training Job** page, set required parameters and click **Submit**.

- **Created By:** Custom algorithms
 - **Boot Mode:** Custom images
 - Image path: image created in [Step 5 Uploading an Image to SWR](#).
 - **Code Directory:** directory where the boot script file is stored in OBS, for example, `obs://test-modelarts/pytorch/demo-code/`. The training code is automatically downloaded to the `${MA_JOB_DIR}/demo-code` directory of the training container. **demo-code** (customizable) is the last-level directory of the OBS path.
 - **Boot Command:** `/home/ma-user/miniconda3/bin/python ${MA_JOB_DIR}/demo-code/pytorch-verification.py`. **demo-code** (customizable) is the last-level directory of the OBS path.
 - **Resource Pool:** Public resource pools
 - **Resource Type:** Select GPU or CPU.
 - **Persistent Log Saving:** enabled
 - **Job Log Path:** Set this parameter to the OBS path for storing training logs, for example, `obs://test-modelarts/pytorch/log/`.
4. Check the parameters of the training job and click **Submit**.
 5. Wait until the training job is completed.

After a training job is created, the operations such as container image downloading, code directory downloading, and boot command execution are automatically performed in the backend. Generally, the training duration ranges from dozens of minutes to several hours, depending on the training procedure and selected resources. After the training job is executed, the log similar to the following is output.

Figure 9-18 Run logs of training jobs with GPU specifications

```
1 tensor([[ -0.4181,  0.8150, -0.2581],
2         [ -0.6062,  0.5347,  0.1890],
3         [ 0.5751,  1.2730, -0.3907],
4         [ 0.4812, -0.4064, -0.2753],
5         [ 1.0377, -1.1248,  1.2977]])
6 tensor([[ -0.7440, -0.8577, -0.2340],
7         [ 0.9569,  0.5516, -1.3350],
8         [-1.2878, -0.2791,  0.3486],
9         [-1.0997,  0.7627, -0.3188],
10        [-1.0865, -1.2626, -0.5900]], device='cuda:0')
11
```

9.4.6 Creating a Custom Training Image (MPI + CPU/GPU)

This section describes how to create an image and use the image for training on the ModelArts platform. The AI engine used for training is MPI, and the resources are CPUs or GPUs.

NOTE

This section applies only to training jobs of the new version.

Scenario

In this example, write a Dockerfile to create a custom image on a Linux x86_64 server running Ubuntu 18.04.

Objective: Build and install container images of the following software and use the images and CPUs/GPUs for training jobs on ModelArts.

- Ubuntu 18.04
- CUDA 11.1
- Python 3.7.13
- Open MPI 3.0.0

Procedure

Before using a custom image to create a training job, get familiar with Docker and have development experience. The following is the detailed procedure:

1. [Prerequisites](#)
2. [Step 1 Creating an OBS Bucket and Folder](#)
3. [Step 2 Preparing Script Files and Uploading Them to OBS](#)
4. [Step 3 Preparing an Image Server](#)
5. [Step 4 Creating a Custom Image](#)
6. [Step 5 Uploading an Image to SWR](#)
7. [Step 6 Creating a Training Job on ModelArts](#)

Prerequisites

You have registered a Huawei ID and enabled Huawei Cloud services, and the account is not in arrears or frozen.

Step 1 Creating an OBS Bucket and Folder

Create a bucket and the folders in OBS for storing the sample dataset and training code. [Table 9-45](#) lists the folders to be created. Replace the bucket and folder names in the example with the actual names.

For details about how to create an OBS bucket and folder, see [Creating a Bucket](#) and [Creating a Folder](#).

Ensure that the OBS directory you use and ModelArts are in the same region.

Table 9-45 Required OBS folders

Folder	Usage
obs://test-modelarts/mpi/demo-code/	Stores the MPI boot script and training script file.
obs://test-modelarts/mpi/log/	Stores training log files.

Step 2 Preparing Script Files and Uploading Them to OBS

Prepare the MPI boot script **run_mpi.sh** and training script **mpi-verification.py** and upload them to the **obs://test-modelarts/mpi/demo-code/** folder of the OBS bucket.

- The content of the MPI boot script **run_mpi.sh** is as follows:

```
#!/bin/bash
MY_HOME=/home/ma-user

MY_SSHD_PORT=${MY_SSHD_PORT:-"38888"}

MY_TASK_INDEX=${MA_TASK_INDEX:-${VC_TASK_INDEX:-${VK_TASK_INDEX}}}

MY_MPI_SLOTS=${MY_MPI_SLOTS:-"${MA_NUM_GPUS}"}

MY_MPI_TUNE_FILE="${MY_HOME}/env_for_user_process"

if [ -z ${MY_MPI_SLOTS} ]; then
    echo "[run_mpi] MY_MPI_SLOTS is empty, set it be 1"
    MY_MPI_SLOTS="1"
fi

printf "MY_HOME: ${MY_HOME}\nMY_SSHD_PORT: ${MY_SSHD_PORT}\nMY_MPI_BTL_TCP_IF: ${MY_MPI_BTL_TCP_IF}\nMY_TASK_INDEX: ${MY_TASK_INDEX}\nMY_MPI_SLOTS: ${MY_MPI_SLOTS}\n"

env | grep -E '^MA_|^SHARED_|^S3_|^PATH|^VC_WORKER|^SCC|^CRED' | grep -v '=' > ${MY_MPI_TUNE_FILE}
# add -x to each line
sed -i 's/^/-x /' ${MY_MPI_TUNE_FILE}

sed -i "s|${MY_SSHD_PORT}|${MY_SSHD_PORT}|g" ${MY_HOME}/etc/ssh/ssh_config

# start sshd service
bash -c "$(which sshd) -f ${MY_HOME}/etc/ssh/ssh_config"

# confirm the sshd is up
netstat -anp | grep LIS | grep ${MY_SSHD_PORT}

if [ $MY_TASK_INDEX -eq 0 ]; then
    # generate the hostfile of mpi
    for ((i=0; i<${MA_NUM_HOSTS}; i++))
    do
        eval hostname=${MA_VJ_NAME}-${MA_TASK_NAME}-${i}.${MA_VJ_NAME}
        echo "[run_mpi] hostname: ${hostname}"

        ip=""
        while [ -z "$ip" ]; do
            ip=$(ping -c 1 ${hostname} | grep "PING" | sed -E 's/PING .*([0-9.]+). */\1/g')
            sleep 1
        done
        echo "[run_mpi] resolved ip: ${ip}"

        # test the sshd is up
        while :
        do
            if [ cat < /dev/null > /dev/tcp/${ip}/${MY_SSHD_PORT} ]; then
                break
            fi
            sleep 1
        done

        echo "[run_mpi] the sshd of ip ${ip} is up"

        echo "${ip} slots=${MY_MPI_SLOTS}" >> ${MY_HOME}/hostfile
    done

    printf "[run_mpi] hostfile:\n`cat ${MY_HOME}/hostfile`\n"
```



```
fi
RET_CODE=0
if [ $MY_TASK_INDEX -eq 0 ]; then
    echo "[run_mpi] start exec command time: "$(date +"%Y-%m-%d-%H:%M:%S")
    np=$(( ${MA_NUM_HOSTS} * ${MY_MPI_SLOTS} ))
    echo "[run_mpi] command: mpirun -np ${np} -hostfile ${MY_HOME}/hostfile -mca plm_rsh_args \"-p ${MY_SSHD_PORT}\" -tune ${MY_MPI_TUNE_FILE} ... @$@"

    # execute mpirun at worker-0
    # mpirun
    mpirun \
        -np ${np} \
        -hostfile ${MY_HOME}/hostfile \
        -mca plm_rsh_args "-p ${MY_SSHD_PORT}" \
        -tune ${MY_MPI_TUNE_FILE} \
        -bind-to none -map-by slot \
        -x NCCL_DEBUG -x NCCL_SOCKET_IFNAME -x NCCL_IB_HCA -x NCCL_IB_TIMEOUT -x
        NCCL_IB_GID_INDEX -x NCCL_IB_TC \
        -x HOROVOD_MPL_THREADS_DISABLE=1 \
        -x PATH -x LD_LIBRARY_PATH \
        -mca pml ob1 -mca btl ^openib -mca plm_rsh_no_tree_spawn true \
        "$@"

    RET_CODE=$?

    if [ $RET_CODE -ne 0 ]; then
        echo "[run_mpi] exec command failed, exited with $RET_CODE"
    else
        echo "[run_mpi] exec command successfully, exited with $RET_CODE"
    fi

    # stop 1...N worker by killing the sleep proc
    sed -i '1d' ${MY_HOME}/hostfile
    if [ `cat ${MY_HOME}/hostfile | wc -l` -ne 0 ]; then
        echo "[run_mpi] stop 1 to (N - 1) worker by killing the sleep proc"

        sed -i 's/${MY_MPI_SLOTS}/1/g' ${MY_HOME}/hostfile
        printf "[run_mpi] hostfile:\n`cat ${MY_HOME}/hostfile`\n"

        mpirun \
            --hostfile ${MY_HOME}/hostfile \
            --mca plm_rsh_args "-p ${MY_SSHD_PORT}" \
            -x PATH -x LD_LIBRARY_PATH \
            pkill sleep \
            > /dev/null 2>&1
    fi

    echo "[run_mpi] exit time: "$(date +"%Y-%m-%d-%H:%M:%S")
else
    echo "[run_mpi] the training log is in worker-0"
    sleep 365d
    echo "[run_mpi] exit time: "$(date +"%Y-%m-%d-%H:%M:%S")
fi
exit $RET_CODE
```

 NOTE

The script **run_mpi.sh** uses LF line endings. If CRLF line endings are used, executing the training job will fail, and the error "\$'\r': command not found" will be displayed in logs.

- The content of the training script **mpi-verification.py** is as follows:

```
import os
import socket
```

```
if __name__ == '__main__':  
    print(socket.gethostname())  
  
# https://www.open-mpi.org/faq/?category=running#mpi-environmental-variables  
print('OMPI_COMM_WORLD_SIZE: ' + os.environ['OMPI_COMM_WORLD_SIZE'])  
print('OMPI_COMM_WORLD_RANK: ' + os.environ['OMPI_COMM_WORLD_RANK'])  
print('OMPI_COMM_WORLD_LOCAL_RANK: ' + os.environ['OMPI_COMM_WORLD_LOCAL_RANK'])
```

Step 3 Preparing an Image Server

Obtain a Linux x86_64 server running Ubuntu 18.04. Either an ECS or your local PC will do.

For details about how to purchase an ECS, see [Purchasing and Logging In to a Linux ECS](#). Set **CPU Architecture** to **x86** and **Image** to **Public image**. Ubuntu 18.04 images are recommended.

Step 4 Creating a Custom Image

Create a container image with the following configurations and use the image to create a training job on ModelArts:

- Ubuntu 18.04
- CUDA 11.1
- Python 3.7.13
- Open MPI 3.0.0

This section describes how to write a Dockerfile to create a custom image.

1. Install Docker.

The following uses the Linux x86_64 OS as an example to describe how to obtain the Docker installation package. For more details, see [Docker official documents](#). Run the following commands to install Docker:

```
curl -fsSL get.docker.com -o get-docker.sh  
sh get-docker.sh
```

If the **docker images** command is executed, Docker has been installed. In this case, skip this step.

2. Check the Docker engine version. Run the following command:

```
docker version | grep -A 1 Engine
```

The following information is displayed:

```
Engine:  
Version:      18.09.0
```

NOTE

Use the Docker engine of the preceding version or later to create a custom image.

3. Create a folder named **context**.

```
mkdir -p context
```

4. Download the Miniconda3 installation file.

Download the Miniconda3 py37 4.12.0 installation file (Python 3.7.13) from https://repo.anaconda.com/miniconda/Miniconda3-py37_4.12.0-Linux-x86_64.sh.

5. Download the openmpi 3.0.0 installation file.

Download the openmpi 3.0.0 file edited using Horovod v0.22.1 from <https://github.com/horovod/horovod/files/1596799/openmpi-3.0.0-bin.tar.gz>.

6. Store the Miniconda3 and openmpi 3.0.0 files in the **context** folder. The following shows the **context** folder:

```
context
├── Miniconda3-py37_4.12.0-Linux-x86_64.sh
└── openmpi-3.0.0-bin.tar.gz
```

7. Write the Dockerfile of the container image.

Create an empty file named **Dockerfile** in the **context** folder and copy the following content to the file:

```
# The host must be connected to the public network for creating a container image.

# Basic container image at https://github.com/NVIDIA/nvidia-docker/wiki/CUDA
#
# https://docs.docker.com/develop/develop-images/multistage-build/#use-multi-stage-builds
# require Docker Engine >= 17.05
#
# builder stage
FROM nvidia/cuda:11.1.1-runtime-ubuntu18.04 AS builder

# The default user of the basic container image is root.
# USER root

# Copy the Miniconda3 (Python 3.7.13) installation files to the /tmp directory of the basic container image.
COPY Miniconda3-py37_4.12.0-Linux-x86_64.sh /tmp

# Install Miniconda3 to the /home/ma-user/miniconda3 directory of the basic container image.
# https://conda.io/projects/conda/en/latest/user-guide/install/linux.html#installing-on-linux
RUN bash /tmp/Miniconda3-py37_4.12.0-Linux-x86_64.sh -b -p /home/ma-user/miniconda3

# Create the final container image.
FROM nvidia/cuda:11.1.1-runtime-ubuntu18.04

# Install vim, curl, net-tools, and the SSH tool in Huawei Mirrors.
RUN cp -a /etc/apt/sources.list /etc/apt/sources.list.bak && \
    sed -i "s@http://.*archive.ubuntu.com@http://repo.huaweicloud.com@g" /etc/apt/sources.list && \
    sed -i "s@http://.*security.ubuntu.com@http://repo.huaweicloud.com@g" /etc/apt/sources.list && \
    echo > /etc/apt/apt.conf.d/00skip-verify-peer.conf "Acquire { https::Verify-Peer false }" && \
    apt-get update && \
    apt-get install -y vim curl net-tools iputils-ping \
    openssh-client openssh-server && \
    ssh -V && \
    mkdir -p /run/sshd && \
    apt-get clean && \
    mv /etc/apt/sources.list.bak /etc/apt/sources.list && \
    rm /etc/apt/apt.conf.d/00skip-verify-peer.conf

# Install the Open MPI 3.0.0 file written using Horovod v0.22.1.
# https://github.com/horovod/horovod/blob/v0.22.1/docker/horovod/Dockerfile
# https://github.com/horovod/horovod/files/1596799/openmpi-3.0.0-bin.tar.gz
COPY openmpi-3.0.0-bin.tar.gz /tmp
RUN cd /usr/local && \
    tar -zxf /tmp/openmpi-3.0.0-bin.tar.gz && \
    ldconfig && \
    mpirun --version

# Add user ma-user (UID = 1000, GID = 100).
# A user group whose GID is 100 of the basic container image exists. User ma-user can directly use it.
RUN useradd -m -d /home/ma-user -s /bin/bash -g 100 -u 1000 ma-user

# Copy the /home/ma-user/miniconda3 directory from the builder stage to the directory with the same name in the current container image.
COPY --chown=ma-user:100 --from=builder /home/ma-user/miniconda3 /home/ma-user/miniconda3

# Configure the preset environment variables of the container image.
```

```
# Set PYTHONUNBUFFERED to 1 to avoid log loss.
ENV PATH=$PATH:/home/ma-user/miniconda3/bin \
  PYTHONUNBUFFERED=1

# Set the default user and working directory of the container image.
USER ma-user
WORKDIR /home/ma-user

# Configure sshd to support SSH password-free login.
RUN MA_HOME=/home/ma-user && \
  # setup sshd dir
  mkdir -p ${MA_HOME}/etc && \
  ssh-keygen -f ${MA_HOME}/etc/ssh_host_rsa_key -N "" -t rsa && \
  mkdir -p ${MA_HOME}/etc/ssh ${MA_HOME}/var/run && \
  # setup sshd config (listen at ${MY_SSHD_PORT} port)
  echo "Port ${MY_SSHD_PORT}"\n\
  HostKey ${MA_HOME}/etc/ssh_host_rsa_key\n\
  AuthorizedKeysFile ${MA_HOME}/.ssh/authorized_keys\n\
  PidFile ${MA_HOME}/var/run/sshd.pid\n\
  StrictModes no\n\
  UsePAM no" > ${MA_HOME}/etc/ssh/sshd_config && \
  # generate ssh key
  ssh-keygen -t rsa -f ${MA_HOME}/.ssh/id_rsa -P "" && \
  cat ${MA_HOME}/.ssh/id_rsa.pub >> ${MA_HOME}/.ssh/authorized_keys && \
  # disable ssh host key checking for all hosts
  echo "Host *\n\
  StrictHostKeyChecking no" > ${MA_HOME}/.ssh/config
```

For details about how to write a Dockerfile, see [official Docker documents](#).

8. Verify that the Dockerfile has been created. The following shows the **context** folder:

```
context
├── Dockerfile
├── Miniconda3-py37_4.12.0-Linux-x86_64.sh
└── openmpi-3.0.0-bin.tar.gz
```

9. Create the container image. Run the following command in the directory where the Dockerfile is stored to build the container image **mpi:3.0.0-cuda11.1**:

```
docker build . -t mpi:3.0.0-cuda11.1
```

The following log information displayed during image creation indicates that the image has been created.

```
naming to docker.io/library/mpi:3.0.0-cuda11.1
```

Step 5 Uploading an Image to SWR

1. Log in to the SWR console and select a region. It must share the same region with ModelArts. Otherwise, the image cannot be selected.
2. Click **Create Organization** in the upper right corner and enter an organization name. Customize the organization name. Replace the organization name **deep-learning** in subsequent commands with the actual organization name.
3. Click **Generate Login Command** in the upper right corner to obtain the login command. In this example, the temporary login command is copied.
4. Log in to the local environment as user **root** and enter the copied temporary login command.
5. Upload the image to SWR.

- a. Run the following command to tag the uploaded image:

```
#Replace the region and domain information with the actual values, and replace the
organization name deep-learning with your custom value.
```

```
sudo docker tag mpi:3.0.0-cuda11.1 swr.cn-north-4.myhuaweicloud.com/deep-learning/mpi:3.0.0-cuda11.1
```

- b. Run the following command to upload the image:

#Replace the region and domain information with the actual values, and replace the organization name **deep-learning** with your custom value.

```
sudo docker push swr.cn-north-4.myhuaweicloud.com/deep-learning/mpi:3.0.0-cuda11.1
```

6. After the image is uploaded, choose **My Images** on the left navigation pane of the SWR console to view the uploaded custom images.

swr.cn-north-4.myhuaweicloud.com/deep-learning/mpi:3.0.0-cuda11.1 is the SWR URL of the custom image.

Step 6 Creating a Training Job on ModelArts

1. Log in to the ModelArts management console, check whether access authorization has been configured for your account. For details, see [Configuring Agency Authorization for ModelArts with One Click](#). If you have been authorized using access keys, clear the authorization and configure agency authorization.
2. Log in to the ModelArts management console. In the navigation pane on the left, choose **Model Training** > **Training Jobs**.
3. On the **Create Training Job** page, configure parameters and click **Submit**.
 - **Created By:** Custom algorithms
 - **Boot Mode:** Custom images
 - **Image path:** **swr.cn-north-4.myhuaweicloud.com/deep-learning/mpi:3.0.0-cuda11.1**
 - **Code Directory:** OBS path to the boot script, for example, **obs://test-modelarts/mpi/demo-code/**.
 - **Boot Command:** **bash \${MA_JOB_DIR}/demo-code/run_mpi.sh python \${MA_JOB_DIR}/demo-code/mpi-verification.py**
 - **Environment Variable:** Add **MY_SSHD_PORT = 38888**.
 - **Resource Pool:** Public resource pools
 - **Compute Nodes:** Enter **1** or **2**.
 - **Persistent Log Saving:** enabled
 - **Job Log Path:** Set this parameter to the OBS path for storing training logs, for example, **obs://test-modelarts/mpi/log/**.
4. Check the parameters of the training job and click **Submit**.
5. Wait until the training job is completed.

After a training job is created, the operations such as container image downloading, code directory downloading, and boot command execution are automatically performed in the backend. Generally, the training duration ranges from dozens of minutes to several hours, depending on the training procedure and selected resources. [Figure 9-19](#) shows the log information after the training job is executed.

Figure 9-19 Run logs of worker-0 with one compute node and GPU specifications

```

MY_HOME: /home/ma-user
MY_SSHD_PORT: 38888
MY_MPI_BTL_TCP_IF: eth0,bond0
MY_TASK_INDEX: 0
MY_MPI_SLOTS: 1
tcp        0      0 0.0.0.0:38888      0.0.0.0:*          LISTEN     60/sshd
tcp6       0      0 :::38888           :::*               LISTEN     60/sshd
172.16.0.122 slots=1
modelarts-job-8cf8a682-21cb-4d73-9bb3-789cecdc458b-worker-0
OMPI_COMM_WORLD_SIZE: 1
OMPI_COMM_WORLD_RANK: 0
OMPI_COMM_WORLD_LOCAL_RANK: 0

```

Set **Compute Nodes** to 2 and run the training job. [Figure 9-20](#) and [Figure 9-21](#) show the log information.

Figure 9-20 Run logs of worker-0 with two compute nodes and GPU specifications

```

MY_HOME: /home/ma-user
MY_SSHD_PORT: 38888
MY_MPI_BTL_TCP_IF: eth0,bond0
MY_TASK_INDEX: 0
MY_MPI_SLOTS: 1
tcp        0      0 0.0.0.0:38888      0.0.0.0:*          LISTEN     61/sshd
tcp6       0      0 :::38888           :::*               LISTEN     61/sshd
172.16.0.39 slots=1
172.16.0.123 slots=1
Warning: Permanently added '[172.16.0.123]:38888' (RSA) to the list of known hosts.
modelarts-job-31732752-6857-4e33-96ff-7a28afae26fb-worker-0
OMPI_COMM_WORLD_SIZE: 2
OMPI_COMM_WORLD_RANK: 0
OMPI_COMM_WORLD_LOCAL_RANK: 0
modelarts-job-31732752-6857-4e33-96ff-7a28afae26fb-worker-1
OMPI_COMM_WORLD_SIZE: 2
OMPI_COMM_WORLD_RANK: 1
OMPI_COMM_WORLD_LOCAL_RANK: 0

```

Figure 9-21 Run logs of worker-1 with two compute nodes and GPU specifications

```

MY_HOME: /home/ma-user
MY_SSHD_PORT: 38888
MY_MPI_BTL_TCP_IF: eth0,bond0
MY_TASK_INDEX: 1
MY_MPI_SLOTS: 1
tcp        0      0 0.0.0.0:38888      0.0.0.0:*          LISTEN     62/sshd
tcp6       0      0 :::38888           :::*               LISTEN     62/sshd
/home/ma-user/modelarts/user-job-dir/xxx/run_mpi.sh: line 109: 66 Terminated          sleep 365d

```

9.4.7 Creating a Custom Training Image (Tensorflow + GPU)

This section describes how to create an image and use it for training on ModelArts. The AI engine used in the image is TensorFlow, and the resources used for training are GPUs.

NOTE

This section applies only to training jobs of the new version.

Scenario

In this example, write a Dockerfile to create a custom image on a Linux x86_64 server running Ubuntu 18.04.

Objective: Build and install container images of the following software and use the images and GPUs for training jobs on ModelArts.

- Ubuntu 18.04
- cuda-11.2
- Python 3.7.13
- mlnx ofed-5.4
- tensorflow gpu-2.10.0

Procedure

Before using a custom image to create a training job, you need to be familiar with Docker and have development experience.

1. [Prerequisites](#)
2. [Step 1 Creating an OBS Bucket and Folder](#)
3. [Step 2 Creating a Dataset and Uploading It to OBS](#)
4. [Step 3 Preparing the Training Script and Uploading It to OBS](#)
5. [Step 4 Preparing a Server](#)
6. [Step 5 Creating a Custom Image](#)
7. [Step 6 Uploading the Image to SWR](#)
8. [Step 7 Creating a Training Job on ModelArts](#)

Prerequisites

You have registered a Huawei Cloud account. The account is not in arrears or frozen.

Step 1 Creating an OBS Bucket and Folder

Create a bucket and folders in OBS for storing the sample dataset and training code. [Table 9-46](#) lists the folders to be created. Replace the bucket and folder names in the example with actual names.

For details about how to create an OBS bucket and folder, see [Creating a Bucket](#) and [Creating a Folder](#).

Ensure that the OBS directory you use and ModelArts are in the same region.

Table 9-46 Required OBS folders

Folder	Usage
obs://test-modelarts/tensorflow/code/	Stores the training script.

Folder	Usage
obs://test-modelarts/tensorflow/data/	Stores dataset files.
obs://test-modelarts/tensorflow/log/	Store training log files.

Step 2 Creating a Dataset and Uploading It to OBS

Download **mnist.npz** from <https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz>, and upload it to **obs://test-modelarts/tensorflow/data/** in the OBS bucket.

Step 3 Preparing the Training Script and Uploading It to OBS

Obtain the training script **mnist.py** and upload it to **obs://test-modelarts/tensorflow/code/** in the OBS bucket.

mnist.py is as follows:

```
import argparse
import tensorflow as tf

parser = argparse.ArgumentParser(description='TensorFlow quick start')
parser.add_argument('--data_url', type=str, default="/Data", help='path where the dataset is saved')
args = parser.parse_args()

mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data(args.data_url)
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10)
])

loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)

model.compile(optimizer='adam',
              loss=loss_fn,
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
```

Step 4 Preparing a Server

Obtain a Linux x86_64 server running Ubuntu 18.04. Either an ECS or your local PC will do.

For details about how to purchase an ECS, see [Purchasing and Logging In to a Linux ECS](#). Set **CPU Architecture** to **x86** and **Image** to **Public image**. Ubuntu 18.04 images are recommended.

Step 5 Creating a Custom Image

Create a container image with the following configurations and use the image to create a training job on ModelArts:

- Ubuntu 18.04
- CUDA 11.1
- Python 3.7.13
- mlnx ofed-5.4
- mindspore gpu-1.8.1

The following describes how to create a custom image by writing a Dockerfile.

1. Install Docker.

The following uses the Linux x86_64 OS as an example to describe how to obtain the Docker installation package. For details about how to install Docker, see [official Docker documents](#). Run the following commands to install Docker:

```
curl -fsSL get.docker.com -o get-docker.sh
sh get-docker.sh
```

If the **docker images** command is executed, Docker has been installed. In this case, skip this step.

2. Check the Docker engine version. Run the following command:

```
docker version | grep -A 1 Engine
```

The following information is displayed:

```
Engine:
Version: 18.09.0
```

 NOTE

Use the Docker engine of the preceding version or later to create a custom image.

3. Create a folder named **context**.

```
mkdir -p context
```

4. Obtain the **pip.conf** file. In this example, the pip source provided by Huawei Mirrors is used. The content of **pip.conf** is as follows:

```
[global]
index-url = https://repo.huaweicloud.com/repository/pypi/simple
trusted-host = repo.huaweicloud.com
timeout = 120
```

 NOTE

To obtain **pip.conf**, go to Huawei Mirrors at <https://mirrors.huaweicloud.com/home> and search for **pypi**.

5. Download **tensorflow_gpu-2.10.0-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl**.

Download **tensorflow_gpu-2.10.0-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl** from <https://pypi.org/project/tensorflow-gpu/2.10.0/#files>.

6. Download the Miniconda3 installation file.

Download the Miniconda3 py37 4.12.0 installation file (Python 3.7.13) from https://repo.anaconda.com/miniconda/Miniconda3-py37_4.12.0-Linux-x86_64.sh.

7. Write the container image Dockerfile.

Create an empty file named **Dockerfile** in the **context** folder and copy the following content to the file:

```
# The server on which the container image is created must access the Internet.

# Base container image at https://github.com/NVIDIA/nvidia-docker/wiki/CUDA
#
# https://docs.docker.com/develop/develop-images/multistage-build/#use-multi-stage-builds
# require Docker Engine >= 17.05
#
# builder stage
FROM nvidia/cuda:11.2.2-cudnn8-runtime-ubuntu18.04 AS builder

# The default user of the base container image is root.
# USER root

# Use the PyPI configuration obtained from Huawei Mirrors.
RUN mkdir -p /root/.pip/
COPY pip.conf /root/.pip/pip.conf

# Copy the installation files to the /tmp directory in the base container image.
COPY Miniconda3-py37_4.12.0-Linux-x86_64.sh /tmp
COPY tensorflow_gpu-2.10.0-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl /tmp

# https://conda.io/projects/conda/en/latest/user-guide/install/linux.html#installing-on-linux
# Install Miniconda3 in the /home/ma-user/miniconda3 directory of the base container image.
RUN bash /tmp/Miniconda3-py37_4.12.0-Linux-x86_64.sh -b -p /home/ma-user/miniconda3

# Install the TensorFlow .whl file using default Miniconda3 Python environment /home/ma-user/miniconda3/bin/pip.
RUN cd /tmp && \
    /home/ma-user/miniconda3/bin/pip install --no-cache-dir \
    /tmp/tensorflow_gpu-2.10.0-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl

RUN cd /tmp && \
    /home/ma-user/miniconda3/bin/pip install --no-cache-dir keras==2.10.0

# Create the container image.
FROM nvidia/cuda:11.2.2-cudnn8-runtime-ubuntu18.04

COPY MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64.tgz /tmp

# Install the vim, curl, net-tools, and MLNX_OFED tools obtained from Huawei Mirrors.
RUN cp -a /etc/apt/sources.list /etc/apt/sources.list.bak && \
    sed -i "s@http://.*archive.ubuntu.com@http://repo.huaweicloud.com@g" /etc/apt/sources.list && \
    sed -i "s@http://.*security.ubuntu.com@http://repo.huaweicloud.com@g" /etc/apt/sources.list && \
    echo > /etc/apt/apt.conf.d/00skip-verify-peer.conf "Acquire { https::Verify-Peer false }" && \
    apt-get update && \
    apt-get install -y vim curl net-tools iputils-ping && \
    # mlnx ofed
    apt-get install -y python libfuse2 dpatch libnl-3-dev autoconf libnl-route-3-dev pciutils libnuma1
libpci3 m4 libelf1 debhelper automake graphviz bison lsof kmod libusb-1.0-0 swig libmnl0 autotools-
dev flex chrpath libltdl-dev && \
    cd /tmp && \
    tar -xvf MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64.tgz && \
    MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64/mlnxofedinstall --user-space-only --basic --
without-fw-update -q && \
    cd - && \
    rm -rf /tmp/* && \
    apt-get clean && \
    mv /etc/apt/sources.list.bak /etc/apt/sources.list && \
    rm /etc/apt/apt.conf.d/00skip-verify-peer.conf

# Add user ma-user (UID = 1000, GID = 100).
# A user group whose GID is 100 exists in the base container image. User ma-user can directly run
the following command:
RUN useradd -m -d /home/ma-user -s /bin/bash -g 100 -u 1000 ma-user
```

```
# Copy the /home/ma-user/miniconda3 directory from the builder stage to the directory with the
same name in the current container image.
COPY --chown=ma-user:100 --from=builder /home/ma-user/miniconda3 /home/ma-user/miniconda3

# Configure the default user and working directory of the container image.
USER ma-user
WORKDIR /home/ma-user

# Configure the preset environment variables of the container image.
# Set PYTHONUNBUFFERED to 1 to prevent log loss.
ENV PATH=/home/ma-user/miniconda3/bin:$PATH \
    LD_LIBRARY_PATH=/usr/local/cuda/lib64:/usr/lib/x86_64-linux-gnu:$LD_LIBRARY_PATH \
    PYTHONUNBUFFERED=1
```

For details about how to write a Dockerfile, see [official Docker documents](#).

8. Download **MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64.tgz**.
Go to [Linux Drivers](#). In the **Download** tab, set **Version** to **5.4-3.5.8.0-LTS**, **OS Distribution Version** to **Ubuntu 18.04**, **Architecture** to **x86_64**, and download **MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64.tgz**.

9. Store the Dockerfile and Miniconda3 installation file in the **context** folder, which is as follows:

```
context
├── Dockerfile
├── MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64.tgz
├── Miniconda3-py37_4.12.0-Linux-x86_64.sh
├── pip.conf
└── tensorflow_gpu-2.10.0-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
```

10. Create the container image. Run the following command in the directory where the Dockerfile is stored to build the container image

```
tensorflow:2.10.0-ofed-cuda11.2:
docker build . -t tensorflow:2.10.0-ofed-cuda11.2
```

The following log shows that the image has been created.
Successfully tagged tensorflow:2.10.0-ofed-cuda11.2

Step 6 Uploading the Image to SWR

1. Log in to the SWR console and select a region. It must share the same region with ModelArts. Otherwise, the image cannot be selected.
2. Click **Create Organization** in the upper right corner and enter an organization name. Customize the organization name. Replace the organization name **deep-learning** in subsequent commands with the actual organization name.
3. Click **Generate Login Command** in the upper right corner to obtain the login command. In this example, the temporary login command is copied.
4. Log in to the local environment as user **root** and enter the copied temporary login command.
5. Upload the image to SWR.

- a. Tag the uploaded image.

```
# Replace the region, domain, as well as organization name deep-learning with the actual values.
sudo docker tag tensorflow:2.10.0-ofed-cuda11.2 swr.{region-id}.{domain}/deep-learning/tensorflow:2.10.0-ofed-cuda11.2
```

- b. Run the following command to upload the image:

```
# Replace the region, domain, as well as organization name deep-learning with the actual values.
sudo docker push swr.{region-id}.{domain}/deep-learning/tensorflow:2.10.0-ofed-cuda11.2
```

6. After the image is uploaded, choose **My Images** in navigation pane on the left of the SWR console to view the uploaded custom images.

Step 7 Creating a Training Job on ModelArts

1. Log in to the ModelArts management console, check whether access authorization has been configured for your account. For details, see [Configuring Agency Authorization for ModelArts with One Click](#). If you have been authorized using access keys, clear the authorization and configure agency authorization.
2. In the navigation pane on the left, choose **Model Training > Training Jobs**. The training job list is displayed by default.
3. Click **Create Training Job**. On the page that is displayed, configure parameters and click **Next**.
 - **Created By:** Custom algorithms
 - **Boot Mode:** Custom images
 - **Image path:** image created in [Step 5 Creating a Custom Image](#) .
 - **Code Directory:** directory where the boot script file is stored in OBS, for example, `obs://test-modelarts/tensorflow/code/`. The training code is automatically downloaded to the `${MA_JOB_DIR}/code` directory of the training container. `code` (customizable) is the last-level directory of the OBS path.
 - **Boot Command:** `python ${MA_JOB_DIR}/code/mnist.py`. `code` (customizable) is the last-level directory of the OBS path.
 - **Training Input:** Click **Add Training Input**. Enter `data_path` for the name, select the OBS path to `mnist.npz`, for example, `obs://test-modelarts/tensorflow/data/mnist.npz`, and set **Obtained from** to **Hyperparameters**.
 - **Resource Pool:** Select **Public resource pools**.
 - **Resource Type:** Select **GPU**.
 - **Compute Nodes:** Enter **1**.
 - **Persistent Log Saving:** enabled
 - **Job Log Path:** OBS path to stored training logs, for example, `obs://test-modelarts/mindspore-gpu/log/`
4. Confirm the configurations of the training job and click **Submit**.
5. Wait until the training job is created.

After you submit the job creation request, the system will automatically perform operations on the backend, such as downloading the container image and code directory and running the boot command. A training job requires a certain period of time for running. The duration ranges from dozens of minutes to several hours, varying depending on the service logic and selected resources. After the training job is executed, the log similar to the following is output.

Figure 9-22 Run logs of training jobs with GPU specifications

```
0.9767.....
323 1503/1875 [=====>.....] - ETA: 0s - loss: 0.0741 - accuracy:
0.9769.....
324 1533/1875 [=====>.....] - ETA: 0s - loss: 0.0743 - accuracy:
0.9769.....
325 1564/1875 [=====>.....] - ETA: 0s - loss: 0.0746 - accuracy:
0.9768.....
326 1595/1875 [=====>.....] - ETA: 0s - loss: 0.0741 - accuracy:
0.9770.....
327 1624/1875 [=====>.....] - ETA: 0s - loss: 0.0742 - accuracy:
0.9770.....
328 1654/1875 [=====>.....] - ETA: 0s - loss: 0.0745 - accuracy:
0.9770.....
329 1685/1875 [=====>.....] - ETA: 0s - loss: 0.0747 - accuracy:
0.9768.....
330 1716/1875 [=====>.....] - ETA: 0s - loss: 0.0752 - accuracy:
0.9767.....
331 1747/1875 [=====>.....] - ETA: 0s - loss: 0.0755 - accuracy:
0.9767.....
332 1778/1875 [=====>.....] - ETA: 0s - loss: 0.0753 - accuracy:
0.9767.....
333 1809/1875 [=====>.....] - ETA: 0s - loss: 0.0751 - accuracy:
0.9768.....
334 1841/1875 [=====>.....] - ETA: 0s - loss: 0.0753 - accuracy:
0.9767.....
335 1872/1875 [=====>.....] - ETA: 0s - loss: 0.0753 - accuracy:
0.9767.....
336 1875/1875 [=====] - 3s 2ms/step - loss: 0.0752 - accuracy: 0.9767
```

9.4.8 Creating a Custom Training Image (MindSpore + Ascend)

This section describes how to create an Ascend container image from scratch and use the image for training on ModelArts. The AI engine used in the image is MindSpore, and the resources used for training are powered by Ascend in a dedicated resource pool.

Description

Objective: Build and install container images of the following software and use the images and Ascend resources for training jobs on ModelArts.

- Ubuntu 18.04
- CANN 8.0.0 (commercial edition)
- Python 3.7.13
- MindSpore 2.1.1

NOTE

- CANN 8.0.0 and MindSpore 2.1.1 are used in the following examples.
- These examples show how to create an Ascend container image and run the image in a dedicated resource pool with the required Ascend driver or firmware installed.

Procedure

Before using a custom image to create a training job, get familiar with Docker and have development experience. The detailed procedure is as follows:

1. [Step 1 Creating an OBS Bucket and Folder](#)
2. [Step 2 Preparing Script Files and Uploading Them to OBS](#)

3. [Step 3 Creating a Custom Image](#)
4. [Step 4 Uploading the Image to SWR](#)
5. [Step 5 Creating and Debugging a Notebook Instance on ModelArts](#)
6. [Step 6 Creating a Training Job on ModelArts](#)

Constraints

- This example requires the CANN commercial edition. If you do not have permission to download the CANN commercial edition, see other examples for creating a custom image.
- Pay attention to the version mapping between MindSpore and CANN, and between CANN and Ascend driver or firmware. Unmatched versions will lead to a training failure.

Step 1 Creating an OBS Bucket and Folder

Create a bucket and folders in OBS for storing the sample dataset and training code. In this example, create a bucket named **test-modelarts** and folders listed in [Table 9-47](#).

For details about how to create an OBS bucket and folder, see [Creating a Bucket](#) and [Creating a Folder](#).

Ensure that the OBS directory you use and ModelArts are in the same region.

Table 9-47 Required OBS folders

Folder	Description
obs://test-modelarts/ascend/demo-code/	Store the Ascend training script.
obs://test-modelarts/ascend/demo-code/run_ascend/	Store the startup scripts of the Ascend training script.
obs://test-modelarts/ascend/log/	Store training log files.

Step 2 Preparing Script Files and Uploading Them to OBS

1. Prepare the training script **mindspore-verification.py** and Ascend startup scripts (five in total) required in this example.
 - For details about the training script, see [Training the mindspore-verification.py File](#).
 - For details about the following Ascend startup scripts, see [Ascend Startup Scripts](#).
 - i. run_ascend.py
 - ii. common.py
 - iii. rank_table.py
 - iv. manager.py

v. `fmk.py`

NOTE

The `mindspore-verification.py` and `run_ascend.py` scripts are invoked by the **Boot Command** parameter during training job creation. For details, see [Boot Command](#).

The `common.py`, `rank_table.py`, `manager.py`, and `fmk.py` scripts are invoked when the `run_ascend.py` script is running.

2. Upload the training script `mindspore-verification.py` to `obs://test-modelarts/ascend/demo-code/` in the OBS bucket.
3. Upload the five Ascend startup scripts to the `obs://test-modelarts/ascend/demo-code/run_ascend/` folder in the OBS bucket.

Step 3 Creating a Custom Image

The following describes how to create a custom image by writing a Dockerfile.

Create a container image with the following configurations and use the image to create a training job on ModelArts:

- Ubuntu 18.04
- CANN 8.0.0 (commercial edition)
- Python 3.7.13
- MindSpore 2.1.1

NOTE

Pay attention to the version mapping between MindSpore and CANN, and between CANN and Ascend driver or firmware. Unmatched versions will lead to a training failure.

These examples show how to create an Ascend container image and run the image in a dedicated resource pool with the required Ascend driver or firmware installed.

1. Obtain a Linux AArch64 server running Ubuntu 18.04. Either an ECS or your local PC will do.

For details about how to purchase an ECS, see [Purchasing and Using a Linux ECS \(New Edition\)](#). Set **CPU Architecture** to **Kunpeng** and **Image** to **Public image**. Ubuntu 18.04 images are recommended.

2. Install Docker.

The following uses Linux AArch64 as an example to describe how to obtain a Docker installation package. For details about how to install Docker, see [official Docker documents](#). Run the following commands to install Docker:

```
curl -fsSL get.docker.com -o get-docker.sh
sh get-docker.sh
```

If the **docker images** command is executed, Docker has been installed. In this case, skip this step.

Start Docker.

```
systemctl start docker
```

3. Check the Docker engine version:

```
docker version | grep -A 1 Engine
```

The command output is as follows:

```
Engine:
Version:      18.09.0
```

 NOTE

Use the Docker engine of the preceding version or later to create a custom image.

4. Create a folder named **context**.

```
mkdir -p context
```

5. Obtain the **pip.conf** file. In this example, the pip source provided by Huawei Mirrors is used, which is as follows:

```
[global]
index-url = https://repo.huaweicloud.com/repository/pypi/simple
trusted-host = repo.huaweicloud.com
timeout = 120
```

 NOTE

To obtain **pip.conf**, go to Huawei Mirrors at <https://mirrors.huaweicloud.com/home> and search for **pypi**.

6. Obtain the APT source file **Ubuntu-Ports-bionic.list**. In this example, the APT source provided by Huawei Mirrors is used. Run the following commands to obtain the APT source file:

```
wget -O Ubuntu-Ports-bionic.list https://repo.huaweicloud.com/repository/conf/Ubuntu-Ports-bionic.list
```

 NOTE

To obtain the APT source file, go to Huawei Mirrors at <https://mirrors.huaweicloud.com/home> and search for **Ubuntu-Ports**. If "failed: Connection time out." is displayed during the download, you can open the download link in the browser for manual download.

7. Contact technical support to download the **CANN 8.0.0-linux aarch64** and **mindspore-2.1.1-cp37-cp37m-linux_aarch64.whl** installation files.
 - Download the **Ascend-cann-nnae_8.0.0_linux-aarch64.run** file.
 - Download the [mindspore-2.1.1-cp37-cp37m-linux_aarch64.whl](#) file.

 NOTE

ModelArts supports only the commercial CANN edition.

8. Download the Miniconda3 installation file.

Download **Miniconda3-py37-4.10.3** (Python 3.7.10) at https://repo.anaconda.com/miniconda/Miniconda3-py37_4.10.3-Linux-aarch64.sh.

9. Store the pip source file, .run file, .whl file, and Miniconda3 installation file in the **context** folder, which is as follows:

```
context
├── Ascend-cann-nnae_8.0.0_linux-aarch64.run
├── mindspore-2.1.1-cp37-cp37m-linux_aarch64.whl
├── Miniconda3-py37_4.10.3-Linux-aarch64.sh
├── pip.conf
└── Ubuntu-Ports-bionic.list
```

10. Write the Dockerfile of the container image.

Create an empty file named **Dockerfile** in the **context** folder and copy the following content to the file:

```
# The server on which the container image is created must access the Internet.
FROM arm64v8/ubuntu:18.04 AS builder

# The default user of the base container image is root.
# USER root

# Install OS dependencies obtained from Huawei Mirrors.
COPY Ubuntu-Ports-bionic.list /tmp
```



```
RUN cp -a /etc/apt/sources.list /etc/apt/sources.list.bak && \
mv /tmp/Ubuntu-Ports-bionic.list /etc/apt/sources.list && \
echo > /etc/apt/apt.conf.d/00skip-verify-peer.conf "Acquire { https::Verify-Peer false }" && \
apt-get update && \
apt-get install -y \
# utils
ca-certificates vim curl \
# CANN 8.0.0
gcc-7 g++ make cmake zlib1g zlib1g-dev openssl libsqlite3-dev libssl-dev libffi-dev unzip pciutils
net-tools libblas-dev gfortran libblas3 && \
apt-get clean && \
mv /etc/apt/sources.list.bak /etc/apt/sources.list && \
# Grant the write permission of the parent directory of the CANN 8.0.0 installation directory to ma-user.
chmod o+w /usr/local

RUN useradd -m -d /home/ma-user -s /bin/bash -g 100 -u 1000 ma-user

# Configure the default user and working directory of the container image.
USER ma-user
WORKDIR /home/ma-user

# Use the PyPI configuration provided by Huawei Mirrors.
RUN mkdir -p /home/ma-user/.pip/
COPY --chown=ma-user:100 pip.conf /home/ma-user/.pip/pip.conf

# Copy the installation files to the /tmp directory in the base container image.
COPY --chown=ma-user:100 Miniconda3-py37_4.10.3-Linux-aarch64.sh /tmp

# https://conda.io/projects/conda/en/latest/user-guide/install/linux.html#installing-on-linux
# Install Miniconda3 in the /home/ma-user/miniconda3 directory of the base container image.
RUN bash /tmp/Miniconda3-py37_4.10.3-Linux-aarch64.sh -b -p /home/ma-user/miniconda3

ENV PATH=$PATH:/home/ma-user/miniconda3/bin

# Install the CANN 8.0.0 Python dependency package.
RUN pip install numpy~=1.14.3 decorator~=4.4.0 sympy~=1.4 cffi~=1.12.3 protobuf~=3.11.3 \
attrs pyyaml pathlib2 scipy requests psutil absl-py

# Install CANN 8.0.0 in /usr/local/Ascend.
COPY --chown=ma-user:100 Ascend-cann-nnae_8.0.0_linux-aarch64.run /tmp
RUN chmod +x /tmp/Ascend-cann-nnae_8.0.0_linux-aarch64.run && \
/tmp/Ascend-cann-nnae_8.0.0_linux-aarch64.run --install --install-path=/usr/local/Ascend

# Install MindSpore 2.1.1.
COPY --chown=ma-user:100 mindspore-2.1.1-cp37-cp37m-linux_aarch64.whl /tmp
RUN chmod +x /tmp/mindspore-2.1.1-cp37-cp37m-linux_aarch64.whl && \
pip install /tmp/mindspore-2.1.1-cp37-cp37m-linux_aarch64.whl

# Create the container image.
FROM arm64v8/ubuntu:18.04

# Install OS dependencies obtained from Huawei Mirrors.
COPY Ubuntu-Ports-bionic.list /tmp
RUN cp -a /etc/apt/sources.list /etc/apt/sources.list.bak && \
mv /tmp/Ubuntu-Ports-bionic.list /etc/apt/sources.list && \
echo > /etc/apt/apt.conf.d/00skip-verify-peer.conf "Acquire { https::Verify-Peer false }" && \
apt-get update && \
apt-get install -y \
# utils
ca-certificates vim curl \
# CANN 8.0.0
gcc-7 g++ make cmake zlib1g zlib1g-dev openssl libsqlite3-dev libssl-dev libffi-dev unzip pciutils
net-tools libblas-dev gfortran libblas3 && \
apt-get clean && \
mv /etc/apt/sources.list.bak /etc/apt/sources.list

RUN useradd -m -d /home/ma-user -s /bin/bash -g 100 -u 1000 ma-user
```

```
# Copy the directories from the builder stage to the directories with the same name in the current
container image.
COPY --chown=ma-user:100 --from=builder /home/ma-user/miniconda3 /home/ma-user/miniconda3
COPY --chown=ma-user:100 --from=builder /home/ma-user/Ascend /home/ma-user/Ascend
COPY --chown=ma-user:100 --from=builder /home/ma-user/var /home/ma-user/var
COPY --chown=ma-user:100 --from=builder /usr/local/Ascend /usr/local/Ascend

# Configure the preset environment variables of the container image.
# Configure CANN environment variables.
# Configure Ascend driver environment variables.
# Set PYTHONUNBUFFERED to 1 to prevent log loss.
ENV PATH=$PATH:/usr/local/Ascend/nnae/latest/bin:/usr/local/Ascend/nnae/latest/compiler/
cccec_compiler/bin:/home/ma-user/miniconda3/bin \
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/Ascend/driver/lib64:/usr/local/Ascend/driver/
lib64/common:/usr/local/Ascend/driver/lib64/driver:/usr/local/Ascend/nnae/latest/lib64:/usr/local/
Ascend/nnae/latest/lib64/plugin/opskernel:/usr/local/Ascend/nnae/latest/lib64/plugin/nnengine \
PYTHONPATH=$PYTHONPATH:/usr/local/Ascend/nnae/latest/python/site-packages:/usr/local/
Ascend/nnae/latest/opp/built-in/op_impl/ai_core/tbe \
ASCEND_AICPU_PATH=/usr/local/Ascend/nnae/latest \
ASCEND_OPP_PATH=/usr/local/Ascend/nnae/latest/opp \
ASCEND_HOME_PATH=/usr/local/Ascend/nnae/latest \
PYTHONUNBUFFERED=1

# Configure the default user and working directory of the container image.
USER ma-user
WORKDIR /home/ma-user
```

For details about how to write a Dockerfile, see [official Docker documents](#).

11. Verify that the Dockerfile has been created. The following shows the **context** folder:

```
context
├── Ascend-cann-nnae_8.0.0_linux-aarch64.run
├── Dockerfile
├── mindspore-2.1.1-cp37-cp37m-linux_aarch64.whl
├── Miniconda3-py37_4.10.3-Linux-aarch64.sh
├── pip.conf
└── Ubuntu-Ports-bionic.list
```

12. Create the container image. Run the following command in the directory where the Dockerfile is stored to create a container image:

```
docker build . -t mindspore:2.1.1-cann8.0.0
```

The following log shows that the image has been created.

```
Successfully tagged mindspore:2.1.1-cann8.0.0
```

13. Upload the created image to SWR. For details, see [Step 4 Uploading the Image to SWR](#).

Step 4 Uploading the Image to SWR

Upload the created image to SWR so that it can be used to create training jobs on ModelArts.

1. Log in to the SWR console and select a region. It must share the same region with ModelArts. Otherwise, the image cannot be selected.
2. Click **Create Organization** in the upper right corner and enter an organization name to create an organization. Customize the organization name. Replace the organization name **deep-learning** in subsequent commands with the actual organization name.
3. Click **Generate Login Command** in the upper right corner to obtain the login command. In this example, the temporary login command is copied.
4. Log in to the local environment as user **root** and enter the copied temporary login command.

5. Upload the image to SWR.
 - a. Run the **docker tag** command to add tags to the uploaded image:
Replace the region, domain, as well as organization name **deep-learning** with the actual values.

```
sudo docker tag mindspore:2.1.1-cann8.0.0 swr.{region}-{domain}/deep-learning/mindspore:2.1.1-cann8.0.0
```
 - b. Run the following command to upload the image:
Replace the region, domain, as well as organization name **deep-learning** with the actual values.

```
sudo docker push swr.{region}-{domain}/deep-learning/mindspore:2.1.1-cann8.0.0
```
6. After the image is uploaded, choose **My Images** in navigation pane on the left of the SWR console to view the uploaded custom images.

Step 5 Creating and Debugging a Notebook Instance on ModelArts

1. Register the image uploaded to SWR with ModelArts Image Management.
Log in to the ModelArts management console. In the navigation pane on the left, choose **Image Management**. Click **Register** and register the image. The registered image can be used to create notebook instances.
2. Use the custom image to create a notebook instance and debug it. After the debugging is successful, save the image.
 - a. [Create a custom image using a custom image.](#)
 - b. [Save the image.](#)
3. After the image is debugged, [create a training job on ModelArts.](#)

Step 6 Creating a Training Job on ModelArts

1. Log in to the ModelArts management console. In the navigation pane on the left, choose **Model Training** > **Training Jobs**. The training job list is displayed by default.
2. On the **Create Training Job** page, configure parameters and click **Submit**.
 - **Algorithm Type:** Custom algorithm
 - **Boot Mode:** Custom image
 - **Image Path:**
 - **Code Directory:** OBS path to store startup scripts, for example, **obs://test-modelarts/ascend/demo-code/**
 - **Boot Command:** **python \${MA_JOB_DIR}/demo-code/run_ascend/run_ascend.py python \${MA_JOB_DIR}/demo-code/mindspore-verification.py**
 - **Resource Pool:** Dedicated resource pools
 - **Resource Type:** Ascend with the required driver and firmware version
 - **Job Log Path:** OBS path to stored training logs, for example, **obs://test-modelarts/ascend/log/**
3. Confirm the configurations of the training job and click **Submit**.
4. Wait until the training job is created.
After you submit the job creation request, the system will automatically perform operations on the backend, such as downloading the container image and code directory and running the boot command. A training job requires a certain period of time for running. The duration ranges from dozens of

minutes to several hours, depending on the service logic and selected resources. After the training job is executed, logs are displayed, as shown in [Figure 9-23](#).

Figure 9-23 Runtime logs of a training job powered by Ascend resources in a dedicated resource pool

```
75 Ascend Envs
76 -----
77 JOB_ID: modelarts-job-2436291a-8543-4ab8-84ad-2dda8f1e4f5c
78 RANK_TABLE_FILE: /home/ma-user/modelarts/rank_table/jobstart_hcc1.json
79 RANK_SIZE: 1
80 ASCEND_DEVICE_ID: 0
81 DEVICE_ID: 0
82 RANK_ID: 0
83 -----
84 [2. 2. 2. 2.]
85 [2. 2. 2. 2.]
86
87 [[2. 2. 2. 2.]
88 [2. 2. 2. 2.]
89 [2. 2. 2. 2.]]
90
91 [[2. 2. 2. 2.]
92 [2. 2. 2. 2.]
93 [2. 2. 2. 2.]]]
```

Training the mindspore-verification.py File

The **mindspore-verification.py** file contains the following information:

```
import os
import numpy as np
from mindspore import Tensor
import mindspore.ops as ops
import mindspore.context as context

print('Ascend Envs')
print('-----')
print('JOB_ID: ', os.environ['JOB_ID'])
print('RANK_TABLE_FILE: ', os.environ['RANK_TABLE_FILE'])
print('RANK_SIZE: ', os.environ['RANK_SIZE'])
print('ASCEND_DEVICE_ID: ', os.environ['ASCEND_DEVICE_ID'])
print('DEVICE_ID: ', os.environ['DEVICE_ID'])
print('RANK_ID: ', os.environ['RANK_ID'])
print('-----')

context.set_context(device_target="Ascend")
x = Tensor(np.ones([1,3,3,4]).astype(np.float32))
y = Tensor(np.ones([1,3,3,4]).astype(np.float32))

print(ops.add(x, y))
```

Ascend Startup Scripts

- **run_ascend.py**

```
import sys
import os

from common import RunAscendLog
from common import RankTableEnv
```

```
from rank_table import RankTable, RankTableTemplate1, RankTableTemplate2

from manager import FMKManager

if __name__ == '__main__':
    log = RunAscendLog.setup_run_ascend_logger()

    if len(sys.argv) <= 1:
        log.error('there are not enough args')
        sys.exit(1)

    train_command = sys.argv[1:]
    log.info('training command')
    log.info(train_command)

    if os.environ.get(RankTableEnv.RANK_TABLE_FILE_V1) is not None:
        # new format rank table file
        rank_table_path = os.environ.get(RankTableEnv.RANK_TABLE_FILE_V1)
        RankTable.wait_for_available(rank_table_path)
        rank_table = RankTableTemplate1(rank_table_path)
    else:
        # old format rank table file
        rank_table_path_origin = RankTableEnv.get_rank_table_template2_file_path()
        RankTable.wait_for_available(rank_table_path_origin)
        rank_table = RankTableTemplate2(rank_table_path_origin)

    if rank_table.get_device_num() >= 1:
        log.info('set rank table %s env to %s' % (RankTableEnv.RANK_TABLE_FILE,
        rank_table.get_rank_table_path()))
        RankTableEnv.set_rank_table_env(rank_table.get_rank_table_path())
    else:
        log.info('device num < 1, unset rank table %s env' % RankTableEnv.RANK_TABLE_FILE)
        RankTableEnv.unset_rank_table_env()

    instance = rank_table.get_current_instance()
    server = rank_table.get_server(instance.server_id)
    current_instance = RankTable.convert_server_to_instance(server)

    fmk_manager = FMKManager(current_instance)
    fmk_manager.run(rank_table.get_device_num(), train_command)
    return_code = fmk_manager.monitor()

    fmk_manager.destroy()

    sys.exit(return_code)
```

- **common.py**

```
import logging
import os

logo = 'Training'

# Rank Table Constants
class RankTableEnv:
    RANK_TABLE_FILE = 'RANK_TABLE_FILE'

    RANK_TABLE_FILE_V1 = 'RANK_TABLE_FILE_V1_0'

    HCCL_CONNECT_TIMEOUT = 'HCCL_CONNECT_TIMEOUT'

    # jobstart_hccl.json is provided by the volcano controller of Cloud-Container-Engine(CCE)
    HCCL_JSON_FILE_NAME = 'jobstart_hccl.json'

    RANK_TABLE_FILE_DEFAULT_VALUE = '/user/config/%s' % HCCL_JSON_FILE_NAME

    @staticmethod
    def get_rank_table_template1_file_dir():
        parent_dir = os.environ[ModelArts.MA_MOUNT_PATH_ENV]
        return os.path.join(parent_dir, 'rank_table')
```

```
@staticmethod
def get_rank_table_template2_file_path():
    rank_table_file_path = os.environ.get(RankTableEnv.RANK_TABLE_FILE)
    if rank_table_file_path is None:
        return RankTableEnv.RANK_TABLE_FILE_DEFAULT_VALUE

    return os.path.join(os.path.normpath(rank_table_file_path),
RankTableEnv.HCCL_JSON_FILE_NAME)

@staticmethod
def set_rank_table_env(path):
    os.environ[RankTableEnv.RANK_TABLE_FILE] = path

@staticmethod
def unset_rank_table_env():
    del os.environ[RankTableEnv.RANK_TABLE_FILE]

class ModelArts:
    MA_MOUNT_PATH_ENV = 'MA_MOUNT_PATH'
    MA_CURRENT_INSTANCE_NAME_ENV = 'MA_CURRENT_INSTANCE_NAME'
    MA_VJ_NAME = 'MA_VJ_NAME'

    MA_CURRENT_HOST_IP = 'MA_CURRENT_HOST_IP'

    CACHE_DIR = '/cache'

    TMP_LOG_DIR = '/tmp/log/'

    FMK_WORKSPACE = 'workspace'

    @staticmethod
    def get_current_instance_name():
        return os.environ[ModelArts.MA_CURRENT_INSTANCE_NAME_ENV]

    @staticmethod
    def get_current_host_ip():
        return os.environ.get(ModelArts.MA_CURRENT_HOST_IP)

    @staticmethod
    def get_job_id():
        ma_vj_name = os.environ[ModelArts.MA_VJ_NAME]
        return ma_vj_name.replace('ma-job', 'modelarts-job', 1)

    @staticmethod
    def get_parent_working_dir():
        if ModelArts.MA_MOUNT_PATH_ENV in os.environ:
            return os.path.join(os.environ.get(ModelArts.MA_MOUNT_PATH_ENV),
ModelArts.FMK_WORKSPACE)

        return ModelArts.CACHE_DIR

class RunAscendLog:

    @staticmethod
    def setup_run_ascend_logger():
        name = logo
        formatter = logging.Formatter(fmt='[run ascend] %(asctime)s - %(levelname)s - %(message)s')

        handler = logging.StreamHandler()
        handler.setFormatter(formatter)

        logger = logging.getLogger(name)
        logger.setLevel(logging.INFO)
        logger.addHandler(handler)
        logger.propagate = False
        return logger
```

```
@staticmethod
def get_run_ascend_logger():
    return logging.getLogger(logo)
```

- rank_table.py

```
import json
import time
import os

from common import ModelArts
from common import RunAscendLog
from common import RankTableEnv

log = RunAscendLog.get_run_ascend_logger()

class Device:
    def __init__(self, device_id, device_ip, rank_id):
        self.device_id = device_id
        self.device_ip = device_ip
        self.rank_id = rank_id

class Instance:
    def __init__(self, pod_name, server_id, devices):
        self.pod_name = pod_name
        self.server_id = server_id
        self.devices = self.parse_devices(devices)

    @staticmethod
    def parse_devices(devices):
        if devices is None:
            return []
        device_object_list = []
        for device in devices:
            device_object_list.append(Device(device['device_id'], device['device_ip'], ""))

        return device_object_list

    def set_devices(self, devices):
        self.devices = devices

class Group:
    def __init__(self, group_name, device_count, instance_count, instance_list):
        self.group_name = group_name
        self.device_count = int(device_count)
        self.instance_count = int(instance_count)
        self.instance_list = self.parse_instance_list(instance_list)

    @staticmethod
    def parse_instance_list(instance_list):
        instance_object_list = []
        for instance in instance_list:
            instance_object_list.append(
                Instance(instance['pod_name'], instance['server_id'], instance['devices']))

        return instance_object_list

class RankTable:
    STATUS_FIELD = 'status'
    COMPLETED_STATUS = 'completed'

    def __init__(self):
        self.rank_table_path = ""
        self.rank_table = {}

    @staticmethod
```

```
def read_from_file(file_path):
    with open(file_path) as json_file:
        return json.load(json_file)

@staticmethod
def wait_for_available(rank_table_file, period=1):
    log.info('Wait for Rank table file at %s ready' % rank_table_file)
    complete_flag = False
    while not complete_flag:
        with open(rank_table_file) as json_file:
            data = json.load(json_file)
            if data[RankTable.STATUS_FIELD] == RankTable.COMPLETED_STATUS:
                log.info('Rank table file is ready for read')
                log.info('\n' + json.dumps(data, indent=4))
                return True

        time.sleep(period)

    return False

@staticmethod
def convert_server_to_instance(server):
    device_list = []
    for device in server['device']:
        device_list.append(
            Device(device_id=device['device_id'], device_ip=device['device_ip'],
rank_id=device['rank_id']))

    ins = Instance(pod_name="", server_id=server['server_id'], devices=[])
    ins.set_devices(device_list)
    return ins

def get_rank_table_path(self):
    return self.rank_table_path

def get_server(self, server_id):
    for server in self.rank_table['server_list']:
        if server['server_id'] == server_id:
            log.info('Current server')
            log.info('\n' + json.dumps(server, indent=4))
            return server

    log.error('server [%s] is not found' % server_id)
    return None

class RankTableTemplate2(RankTable):

    def __init__(self, rank_table_template2_path):
        super().__init__()

        json_data = self.read_from_file(file_path=rank_table_template2_path)

        self.status = json_data[RankTableTemplate2.STATUS_FIELD]
        if self.status != RankTableTemplate2.COMPLETED_STATUS:
            return

        # sorted instance list by the index of instance
        # assert there is only one group
        json_data["group_list"][0]["instance_list"] = sorted(json_data["group_list"][0]["instance_list"],
                                                                key=RankTableTemplate2.get_index)

        self.group_count = int(json_data['group_count'])
        self.group_list = self.parse_group_list(json_data['group_list'])

        self.rank_table_path, self.rank_table = self.convert_template2_to_template1_format_file()

    @staticmethod
    def parse_group_list(group_list):
```



```
group_object_list = []
for group in group_list:
    group_object_list.append(
        Group(group['group_name'], group['device_count'], group['instance_count'],
group['instance_list']))

return group_object_list

@staticmethod
def get_index(instance):
    # pod_name example: job94dc1dbf-job-bj4-yolov4-15
    pod_name = instance["pod_name"]
    return int(pod_name[pod_name.rfind("-") + 1:])

def get_current_instance(self):
    """
    get instance by pod name
    specially, return the first instance when the pod name is None
    :return:
    """
    pod_name = ModelArts.get_current_instance_name()
    if pod_name is None:
        if len(self.group_list) > 0:
            if len(self.group_list[0].instance_list) > 0:
                return self.group_list[0].instance_list[0]

        return None

    for group in self.group_list:
        for instance in group.instance_list:
            if instance.pod_name == pod_name:
                return instance
    return None

def convert_template2_to_template1_format_file(self):
    rank_table_template1_file = {
        'status': 'completed',
        'version': '1.0',
        'server_count': '0',
        'server_list': []
    }

    logic_index = 0
    server_map = {}
    # collect all devices in all groups
    for group in self.group_list:
        if group.device_count == 0:
            continue
        for instance in group.instance_list:
            if instance.server_id not in server_map:
                server_map[instance.server_id] = []

            for device in instance.devices:
                template1_device = {
                    'device_id': device.device_id,
                    'device_ip': device.device_ip,
                    'rank_id': str(logic_index)
                }
                logic_index += 1
                server_map[instance.server_id].append(template1_device)

    server_count = 0
    for server_id in server_map:
        rank_table_template1_file['server_list'].append({
            'server_id': server_id,
            'device': server_map[server_id]
        })
    server_count += 1
```

```
rank_table_template1_file['server_count'] = str(server_count)

log.info('Rank table file (Template1)')
log.info('\n' + json.dumps(rank_table_template1_file, indent=4))

if not os.path.exists(RankTableEnv.get_rank_table_template1_file_dir()):
    os.makedirs(RankTableEnv.get_rank_table_template1_file_dir())

path = os.path.join(RankTableEnv.get_rank_table_template1_file_dir(),
RankTableEnv.HCCL_JSON_FILE_NAME)
with open(path, 'w') as f:
    f.write(json.dumps(rank_table_template1_file))
    log.info('Rank table file (Template1) is generated at %s', path)

return path, rank_table_template1_file

def get_device_num(self):
    total_device_num = 0
    for group in self.group_list:
        total_device_num += group.device_count
    return total_device_num

class RankTableTemplate1(RankTable):
    def __init__(self, rank_table_template1_path):
        super().__init__()
        self.rank_table_path = rank_table_template1_path
        self.rank_table = self.read_from_file(file_path=rank_table_template1_path)

    def get_current_instance(self):
        current_server = None
        server_list = self.rank_table['server_list']
        if len(server_list) == 1:
            current_server = server_list[0]
        elif len(server_list) > 1:
            host_ip = ModelArts.get_current_host_ip()
            if host_ip is not None:
                for server in server_list:
                    if server['server_id'] == host_ip:
                        current_server = server
                        break
            else:
                current_server = server_list[0]

        if current_server is None:
            log.error('server is not found')
            return None
        return self.convert_server_to_instance(current_server)

    def get_device_num(self):
        server_list = self.rank_table['server_list']
        device_num = 0
        for server in server_list:
            device_num += len(server['device'])
        return device_num
```

- **manager.py**

```
import time
import os
import os.path
import signal

from common import RunAscendLog
from fmk import FMK

log = RunAscendLog.get_run_ascend_logger()

class FMKManager:
```

```
# max destroy time: ~20 (15 + 5)
# ~ 15 (1 + 2 + 4 + 8)
MAX_TEST_PROC_CNT = 4

def __init__(self, instance):
    self.instance = instance
    self.fmk = []
    self.fmk_processes = []
    self.get_sigterm = False
    self.max_test_proc_cnt = FMKManager.MAX_TEST_PROC_CNT

# break the monitor and destroy processes when get terminate signal
def term_handle(func):
    def receive_term(signum, stack):
        log.info('Received terminate signal %d, try to destroyed all processes' % signum)
        stack.f_locals['self'].get_sigterm = True

    def handle_func(self, *args, **kwargs):
        origin_handle = signal.getsignal(signal.SIGTERM)
        signal.signal(signal.SIGTERM, receive_term)
        res = func(self, *args, **kwargs)
        signal.signal(signal.SIGTERM, origin_handle)
        return res

    return handle_func

def run(self, rank_size, command):
    for index, device in enumerate(self.instance.devices):
        fmk_instance = FMK(index, device)
        self.fmk.append(fmk_instance)

        self.fmk_processes.append(fmk_instance.run(rank_size, command))

@term_handle
def monitor(self, period=1):
    # busy waiting for all fmk processes exit by zero
    # or there is one process exit by non-zero

    fmk_cnt = len(self.fmk_processes)
    zero_ret_cnt = 0
    while zero_ret_cnt != fmk_cnt:
        zero_ret_cnt = 0
        for index in range(fmk_cnt):
            fmk = self.fmk[index]
            fmk_process = self.fmk_processes[index]
            if fmk_process.poll() is not None:
                if fmk_process.returncode != 0:
                    log.error('proc-rank-%s-device-%s (pid: %d) has exited with non-zero code: %d'
                              % (fmk.rank_id, fmk.device_id, fmk_process.pid, fmk_process.returncode))
                    return fmk_process.returncode

            zero_ret_cnt += 1
        if self.get_sigterm:
            break
        time.sleep(period)

    return 0

def destroy(self, base_period=1):
    log.info('Begin destroy training processes')
    self.send_sigterm_to_fmk_process()
    self.wait_fmk_process_end(base_period)
    log.info('End destroy training processes')

def send_sigterm_to_fmk_process(self):
    # send SIGTERM to fmk processes (and process group)
    for r_index in range(len(self.fmk_processes) - 1, -1, -1):
        fmk = self.fmk[r_index]
        fmk_process = self.fmk_processes[r_index]
```

```
if fmk_process.poll() is not None:
    log.info('proc-rank-%s-device-%s (pid: %d) has exited before receiving the term signal',
            fmk.rank_id, fmk.device_id, fmk_process.pid)
    del self.fmk_processes[r_index]
    del self.fmk[r_index]

try:
    os.killpg(fmk_process.pid, signal.SIGTERM)
except ProcessLookupError:
    pass

def wait_fmk_process_end(self, base_period):
    test_cnt = 0
    period = base_period
    while len(self.fmk_processes) > 0 and test_cnt < self.max_test_proc_cnt:
        for r_index in range(len(self.fmk_processes) - 1, -1, -1):
            fmk = self.fmk[r_index]
            fmk_process = self.fmk_processes[r_index]
            if fmk_process.poll() is not None:
                log.info('proc-rank-%s-device-%s (pid: %d) has exited',
                        fmk.rank_id, fmk.device_id, fmk_process.pid)
                del self.fmk_processes[r_index]
                del self.fmk[r_index]
            if not self.fmk_processes:
                break

        time.sleep(period)
        period *= 2
        test_cnt += 1

    if len(self.fmk_processes) > 0:
        for r_index in range(len(self.fmk_processes) - 1, -1, -1):
            fmk = self.fmk[r_index]
            fmk_process = self.fmk_processes[r_index]
            if fmk_process.poll() is None:
                log.warn('proc-rank-%s-device-%s (pid: %d) has not exited within the max waiting time,
                        'send kill signal',
                        fmk.rank_id, fmk.device_id, fmk_process.pid)
                os.killpg(fmk_process.pid, signal.SIGKILL)
```

- **fmk.py**

```
import os
import subprocess
import pathlib
from contextlib import contextmanager

from common import RunAscendLog
from common import RankTableEnv
from common import ModelArts

log = RunAscendLog.get_run_ascend_logger()

class FMK:

    def __init__(self, index, device):
        self.job_id = ModelArts.get_job_id()
        self.rank_id = device.rank_id
        self.device_id = str(index)

    def gen_env_for_fmk(self, rank_size):
        current_envs = os.environ.copy()

        current_envs['JOB_ID'] = self.job_id

        current_envs['ASCEND_DEVICE_ID'] = self.device_id
        current_envs['DEVICE_ID'] = self.device_id

        current_envs['RANK_ID'] = self.rank_id
```

```
current_envs['RANK_SIZE'] = str(rank_size)

FMK.set_env_if_not_exist(current_envs, RankTableEnv.HCCL_CONNECT_TIMEOUT, str(1800))

log_dir = FMK.get_log_dir()
process_log_path = os.path.join(log_dir, self.job_id, 'ascend', 'process_log', 'rank_' + self.rank_id)
FMK.set_env_if_not_exist(current_envs, 'ASCEND_PROCESS_LOG_PATH', process_log_path)
pathlib.Path(current_envs['ASCEND_PROCESS_LOG_PATH']).mkdir(parents=True, exist_ok=True)

return current_envs

@contextmanager
def switch_directory(self, directory):
    owd = os.getcwd()
    try:
        os.chdir(directory)
        yield directory
    finally:
        os.chdir(owd)

def get_working_dir(self):
    fmk_workspace_prefix = ModelArts.get_parent_working_dir()
    return os.path.join(os.path.normpath(fmk_workspace_prefix), 'device%s' % self.device_id)

@staticmethod
def get_log_dir():
    parent_path = os.getenv(ModelArts.MA_MOUNT_PATH_ENV)
    if parent_path:
        log_path = os.path.join(parent_path, 'log')
        if os.path.exists(log_path):
            return log_path

    return ModelArts.TMP_LOG_DIR

@staticmethod
def set_env_if_not_exist(envs, env_name, env_value):
    if env_name in os.environ:
        log.info('env already exists. env_name: %s, env_value: %s ' % (env_name, env_value))
        return

    envs[env_name] = env_value

def run(self, rank_size, command):
    envs = self.gen_env_for_fmk(rank_size)
    log.info('bootstrap proc-rank-%s-device-%s' % (self.rank_id, self.device_id))

    log_dir = FMK.get_log_dir()
    if not os.path.exists(log_dir):
        os.makedirs(log_dir)

    log_file = '%s-proc-rank-%s-device-%s.txt' % (self.job_id, self.rank_id, self.device_id)
    log_file_path = os.path.join(log_dir, log_file)

    working_dir = self.get_working_dir()
    if not os.path.exists(working_dir):
        os.makedirs(working_dir)

    with self.switch_directory(working_dir):
        # os.setsid: change the process(forked) group id to itself
        training_proc = subprocess.Popen(command, env=envs, preexec_fn=os.setsid,
                                          stdout=subprocess.PIPE, stderr=subprocess.STDOUT)

        log.info('proc-rank-%s-device-%s (pid: %d)', self.rank_id, self.device_id, training_proc.pid)

        # https://docs.python.org/3/library/subprocess.html#subprocess.Popen.wait
        subprocess.Popen(['tee', log_file_path], stdin=training_proc.stdout)

    return training_proc
```

9.5 Creating a Custom Image for Inference

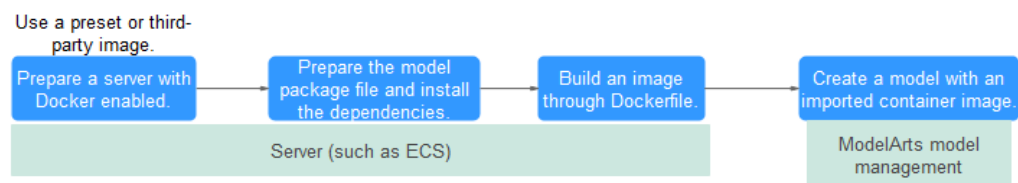
9.5.1 Creating a Custom Image for a Model

If you have developed a model using an AI engine that is not supported by ModelArts, to use this model to create AI applications, create a custom image, import the image to ModelArts, and use it to create models. The models created in this way can be centrally managed and deployed as services.

Procedure

: The preset image does not meet the software environment requirements. You need to import a model package. The new image is larger than 35 GB and needs to be created on a server such as ECS. For details, see [Creating a Custom Image on ECS](#).

Figure 9-24 Creating a custom image for a model



Constraints

- No malicious code is allowed.
- The image for creating a model cannot be larger than 50 GB.
- For models in synchronous request mode, if the prediction request latency exceeds 60 seconds, the request will fail, and there is a possibility that the service may be interrupted. Therefore, in this case, create a model in asynchronous mode.

Specifications for Custom Images

- **External APIs**

Set the external service API for a custom image. The inference API must be the same as the URL defined by **apis** in **config.json**. Then, the external service API can be directly accessed when the image is started. The following is an example of accessing an MNIST image. The image contains a model trained using an MNIST dataset and can identify handwritten digits. **listen_ip** indicates the container IP address. You can start a custom image to obtain the container IP address from the container.

- Sample request

```
curl -X POST \ http://{listen_ip}:8080/ \ -F images=@seven.jpg
```

Figure 9-25 Example of obtaining `listen_ip`

```
root@192.168.1.3:/# cat /etc/hosts
127.0.0.1    localhost
::1         localhost ip6-localhost ip6-loopback
fe00::0     ip6-localnet
ff00::0     ip6-mcastprefix
ff02::1     ip6-allnodes
ff02::2     ip6-allrouters
169.254.30.2 d6211431d0e3
```

- Sample response
`{"mnist_result": 7}`

- **(Optional) Health check APIs**

If services must not be interrupted during a rolling upgrade, the health check APIs must be configured in **config.json** for ModelArts. The health check APIs return the health status for a service when the service is running properly or an error when the service becomes faulty.

NOTICE

- The health check APIs must be configured for a hitless rolling upgrade.
- If you need to use OBS external storage mounting for custom images in real-time services, create a new directory for OBS data, for example, **/obs-mount/**. Otherwise, the existing files will be overwritten. You can add, view, and modify files in the OBS mount directory. To delete the files, delete them in the OBS parallel file system.

The following shows a sample health check API:

- URI
`GET /health`
- Sample request: `curl -X GET \ http://{<listening-IP-address>}:8080/health`
- Sample response
`{"health": "true"}`
- Status code

Table 9-48 Status code

Status Code	Message	Description
200	OK	Request sent.

- **Log file output**

To ensure that logs can be properly displayed, the logs must be standard output.

- **Image boot file**

To deploy a batch service, set the boot file of an image to **/home/run.sh** and use CMD to set the default boot path. The following is a sample Dockerfile:

CMD ["sh", "/home/run.sh"]

- **Image dependencies**

To deploy a batch service, install dependencies such as Python, JRE/JDK, and ZIP in the image.

- **(Optional) Maintaining HTTP persistent connections for hitless rolling upgrade**

To ensure that services are not interrupted during a rolling upgrade, set HTTP **keep-alive** to **200**. For example, Gunicorn does not support keep-alive by default. To ensure hitless rolling upgrade, install Gevent and configure **--keep-alive 200 -k gevent** in the image. The parameter settings vary depending on the service framework. Set the parameters as required.

- **(Optional) Processing SIGTERM signals and gracefully exiting a container**

To ensure that services are not interrupted during a rolling upgrade, the system must capture SIGTERM signals in the container and wait for 60s before gracefully exiting the container. If the duration is less than 60s before the graceful exit, services may be interrupted during the rolling upgrade. To ensure uninterrupted service running, the system exits the container after the system receives SIGTERM signals and processes all received requests. The whole duration is not longer than 90s. The following shows example **run.sh**:

```
#!/bin/bash
gunicorn_pid=""

handle_sigterm() {
    echo "Received SIGTERM, send SIGTERM to $gunicorn_pid"
    if [ $gunicorn_pid != "" ]; then
        sleep 60
        kill -15 $gunicorn_pid # Pass SIGTERM signals to the Gunicorn process.
        wait $gunicorn_pid     # Wait until the Gunicorn process stops.
    fi
}

trap handle_sigterm TERM
```

9.5.2 Creating a Custom Image on ECS

If you want to use an AI engine that is not supported by ModelArts, create a custom image for the engine, import the image to ModelArts, and use the image to create models. This section describes how to use a custom image to create a model and deploy it as a real-time service.

The procedure is as follows:

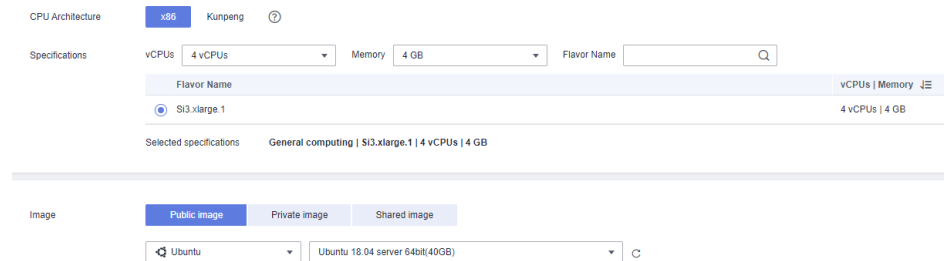
1. **Building an Image Locally**: Create a custom image package locally. For details, see [Creating a Custom Image for a Model](#).
2. **Verifying the Image Locally and Uploading It to SWR**: Verify the APIs of the custom image and upload the custom image to SWR.
3. **Creating a Model Using a Custom Image**: Import the image to ModelArts AI application management.
4. **Deploying the Model as a Real-Time Service**: Deploy the model as a real-time service.

Building an Image Locally

This section uses a Linux x86_x64 host as an example. You can purchase an ECS of the same specifications or use an existing local host to create a custom image.

For details about how to purchase an ECS, see [Purchasing and Logging In to a Linux ECS](#). When creating the ECS, select an Ubuntu 18.04 public image.

Figure 9-26 Creating an ECS using an x86 public image



1. After logging in to the host, install Docker. For details, see [Docker official documents](#). Alternatively, run the following commands to install Docker:

```
curl -fsSL get.docker.com -o get-docker.sh
sh get-docker.sh
```
2. Obtain the base image. Ubuntu 18.04 is used in this example.

```
docker pull ubuntu:18.04
```
3. Create the **self-define-images** folder, and edit **Dockerfile** and **test_app.py** in the folder for the custom image. In the sample code, the application code runs on the Flask framework.

The file structure is as follows:

```
self-define-images/
--Dockerfile
--test_app.py
```

– Dockerfile

```
From ubuntu:18.04
# Configure the Huawei Cloud source and install Python, Python3-PIP, and Flask.
RUN cp -a /etc/apt/sources.list /etc/apt/sources.list.bak && \
  sed -i "s@http://.*security.ubuntu.com@http://repo.huaweicloud.com@g" /etc/apt/sources.list && \
  sed -i "s@http://.*archive.ubuntu.com@http://repo.huaweicloud.com@g" /etc/apt/sources.list && \
  apt-get update && \
  apt-get install -y python3 python3-pip && \
  pip3 install --trusted-host https://repo.huaweicloud.com -i https://repo.huaweicloud.com/
repository/pypi/simple Flask

# Copy the application code to the image.
COPY test_app.py /opt/test_app.py

# Specify the boot command of the image.
CMD python3 /opt/test_app.py
```

– test_app.py

```
from flask import Flask, request
import json
app = Flask(__name__)

@app.route('/greet', methods=['POST'])
def say_hello_func():
    print("----- in hello func -----")
    data = json.loads(request.get_data(as_text=True))
    print(data)
    username = data['name']
    rsp_msg = 'Hello, {}'.format(username)
    return json.dumps({"response":rsp_msg}, indent=4)
```

```
@app.route('/goodbye', methods=['GET'])
def say_goodbye_func():
    print("----- in goodbye func -----")
    return '\nGoodbye!\n'

@app.route('/', methods=['POST'])
def default_func():
    print("----- in default func -----")
    data = json.loads(request.get_data(as_text=True))
    return '\n called default func !\n {} \n'.format(str(data))

# host must be "0.0.0.0", port must be 8080
if __name__ == '__main__':
    app.run(host="0.0.0.0", port=8080)
```

4. Switch to the **self-define-images** folder and run the following command to create custom image **test:v1**:
`docker build -t test:v1 .`
5. Run **docker images** to view the custom image you have created.

Verifying the Image Locally and Uploading It to SWR

1. Run the following command in the local environment to start the custom image:
`docker run -it -p 8080:8080 test:v1`

Figure 9-27 Starting a custom image

2. Open another terminal and run the following commands to test the functions of the three APIs of the custom image:
`curl -X POST -H "Content-Type: application/json" --data '{"name":"Tom"}' 127.0.0.1:8080/`
`curl -X POST -H "Content-Type: application/json" --data '{"name":"Tom"}' 127.0.0.1:8080/greet`
`curl -X GET 127.0.0.1:8080/goodbye`

If information similar to the following is displayed, the function verification is successful.

Figure 9-28 Testing API functions

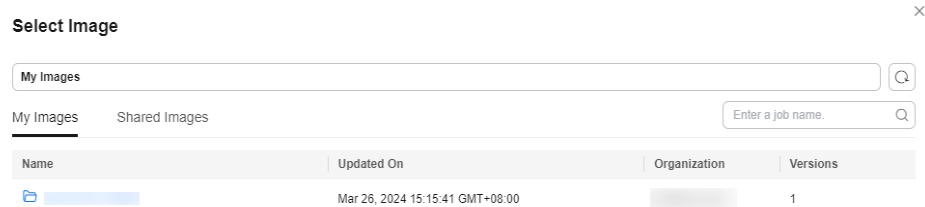
```
root@:~# curl -X POST -H "Content-Type: application/json" --data '{"name":"Tom"}' 127.0.0.1:8080/
{"name": "Tom"}
root@:~# curl -X POST -H "Content-Type: application/json" --data '{"name":"Tom"}' 127.0.0.1:8080/greet
{"response": "Hello, Tom!"}
root@:~# curl -X GET 127.0.0.1:8080/goodbye
Goodbye!
```

3. Upload the custom image to SWR.
4. After the custom image is uploaded, view the uploaded image on the **My Images > Private Images** page of the SWR console.

Creating a Model Using a Custom Image

When you import a meta model by referring to [Importing a Meta Model from a Container Image](#), pay attention to the following parameters:

- **Meta Model Source:** Select **Container image**.
 - **Container Image Path:** Select the created private image.

Figure 9-29 Created private image

- **Container API:** Protocol and port number for starting a model. Ensure that the protocol and port number are the same as those provided in the custom image.
 - **Image Replication:** indicates whether to copy the model image in the container image to ModelArts. This parameter is optional.
 - **Health Check:** checks health status of a model. This parameter is optional. This parameter is configurable only when the health check API is configured in the custom image. Otherwise, creating the model will fail.
- **APIs:** APIs of a custom image. This parameter is optional. Ensure that the APIs comply with ModelArts specifications. For details, see [Specifications for Editing a Model Configuration File](#).

The configuration file is as follows:

```
[{
  "url": "/",
  "method": "post",
  "request": {
    "Content-type": "application/json"
  },
  "response": {
    "Content-type": "application/json"
  }
},
{
  "url": "/greet",
  "method": "post",
  "request": {
    "Content-type": "application/json"
  },
  "response": {
    "Content-type": "application/json"
  }
},
{
  "url": "/goodbye",
  "method": "get",
  "request": {
    "Content-type": "application/json"
  },
  "response": {
    "Content-type": "application/json"
  }
}
```

Deploying the Model as a Real-Time Service

1. [Deploy a model as a real-time service](#).

2. View the details about the real-time service.
3. Access the real-time service in the **Prediction** tab.

10 Monitoring ModelArts Standard Resources

10.1 Overview

ModelArts Standard provides multiple methods for you to view metrics.

- Method 1: View the metrics on the ModelArts Standard console.
You can view the metrics on the ModelArts **Overview** page or the monitoring tab of each module.
 - You can view metrics on the ModelArts **Overview** page. For details, see [Viewing Monitoring Metrics on the ModelArts Console](#).
 - Standard training jobs: When a training job is running, you can view the usage of resources such as CPUs, GPUs, and NPUs. For details, see [Viewing the Resource Usage of a Training Job](#).
 - Standard real-time services: After deploying a model as a real-time service, you can see how much CPU, memory, and GPU resources it is using, as well as how many model calls it has received. For details, see [Viewing Details About a Real-Time Service](#).
- Method 2: View all metrics on AOM.
All metrics reported by ModelArts Standard are stored in AOM, which enables you to consume metrics. You can set metric threshold alarms and report alarms on the AOM console. For details, see [Viewing All ModelArts Monitoring Metrics on the AOM Console](#).
- Method 3: View all metrics on Grafana.
If AOM's monitoring templates cannot meet your requirements, you can use Grafana to view and analyze metrics. Grafana provides different views and templates for monitoring, which allow you to see the real-time resource usage on dashboards clearly.
After adding the data source in Grafana, you can view all ModelArts Standard monitoring metrics stored in AOM using Grafana. For details, see [Using Grafana to View AOM Monitoring Metrics](#).
To view AOM monitoring metrics using Grafana plugins, follow these steps:
 - a. Install and configure Grafana.

You can install and configure Grafana using any of the following ways: [Installing and Configuring Grafana on Windows](#), [Installing and Configuring Grafana on Linux](#), and [Installing and Configuring Grafana on a Notebook Instance](#).

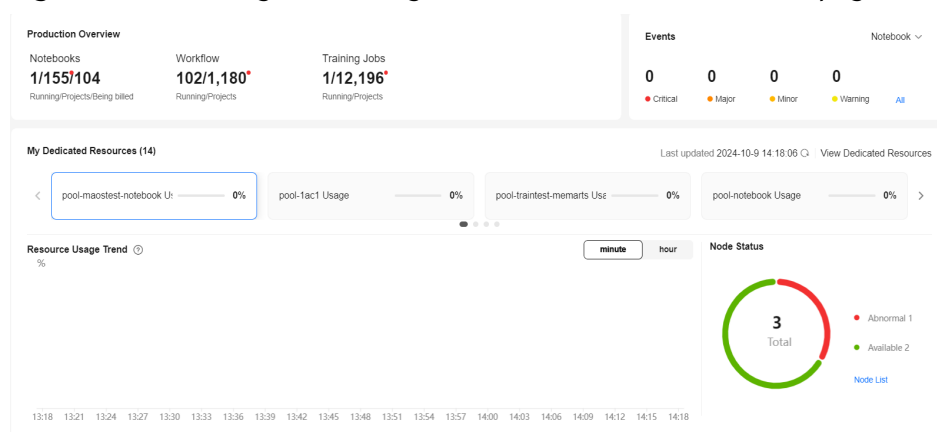
- b. [Configuring a Grafana Data Source](#)
- c. [Configuring a Dashboard to View Metric Data](#)

10.2 Viewing Monitoring Metrics on the ModelArts Console

Viewing ModelArts Monitoring Metrics on the Overview Page

On the ModelArts overview page, you can view various information, including production overview, resource usage, and resource usage of training jobs. Click the links to view more details.

Figure 10-1 Viewing monitoring information on the **Overview** page



NOTE

If there is a discrepancy between the total number of events displayed at the top and bottom of the **Overview** page, refresh the page.

Viewing ModelArts Monitoring Metrics of Each Module

- Training jobs: When a training job is running, you can view the usage of resources such as CPUs, GPUs, and NPUs. For details, see [Viewing the Resource Usage of a Training Job](#).
- Real-time services: After deploying a model as a real-time service, you can see how much CPU, memory, and GPU resources it is using, as well as how many model calls it has received. For details, see [Viewing Details About a Real-Time Service](#).

10.3 Viewing All ModelArts Monitoring Metrics on the AOM Console

ModelArts periodically collects the usage of key metrics (such as GPUs, NPUs, CPUs, and memory) of each node in a resource pool as well as the usage of key metrics of development environments, training jobs, and inference services, and reports the data to AOM. You can view the information on AOM.

Viewing Monitoring Metrics on the AOM Console

1. Log in to the console and search for **AOM** to access the AOM console. In the navigation pane on the left, choose **Monitoring > Metric Monitoring**. Set the metric source to the default instance **Prometheus_AOM_Default** and configure the statistical mode and period as required.
2. At the bottom of the metric browsing page, select one or more target metrics by configuring **All metrics** or **Prometheus statement**.

The following case uses obtaining training job metrics by configuring **All metrics** as an example. Select a specific metric, for example, **ma_container_cpu_util**. Then, configure **Conditions** by setting **Dimension name** to **instance_name** and **Dimension value** to the training job ID obtained on the training details page of the ModelArts console. As a result, the metric monitoring curve of the current training job is displayed.

For details about how to add metrics by configuring **Prometheus statement**, see ["Observability Metric Browsing" in Application Operations Management > User Guide \(2.0\)](#).

For details about more monitoring metrics, see [Table 10-1](#) and [Table 10-2](#).

Container-level Metrics

Table 10-1 Container metrics

Category	Name	Metric	Description	Unit	Value Range	Alarm Threshold	Alarm Severity	Solution
CPU	CPU Usage	ma_container_cpu_util	CPU usage of a measured object	%	0%–100%	Raw data > 95% for two consecutive periods	Suggestion	Check whether the service resource usage meets the expectation. If the service is normal, no action is required.
	Used CPU Cores	ma_container_cpu_used_core	Number of CPU cores used by a measured object	Cores	≥ 0	N/A	N/A	N/A
	Total CPU Cores	ma_container_cpu_limit_core	Total number of CPU cores that have been applied for a measured object	Cores	≥ 1	N/A	N/A	N/A

Category	Name	Metric	Description	Unit	Value Range	Alarm Threshold	Alarm Severity	Solution
	CPU Memory Usage	ma_container_gpu_mem_util	Percentage of the used GPU memory to the total GPU memory	%	0%–100%	Raw data > 95% for two consecutive periods	Suggestion	Check whether the service resource usage meets the expectation. If the service is normal, no action is required.
Memory	Total Physical Memory	ma_container_memory_capacity_megabytes	Total physical memory that has been requested for a measured object	MB	≥ 0	N/A	N/A	N/A

Cate gory	Name	Metric	Description	Unit	Val ue Ran ge	Alar m Thres hold	Al ar m Se ve rit y	Solut ion
	Physical Memory Usage	ma_cont ainer_me memory_util	Percentage of the used physical memory to the total physical memory	%	0%– 100 %	Raw data > 95% for two conse cutiv e perio ds	Su gg est ion	Chec k whet her the servi ce resou rce usag e meet s the expe ctati on. If the servi ce is norm al, no actio n is requi red.

Cate gory	Name	Metric	Description	Unit	Val ue Ran ge	Alar m Thres hold	Al ar m Se ve rit y	Solut ion
	Used Physical Memory	ma_cont ainer_me mory_us ed_mega bytes	Physical memory that has been used by a measured object (container_memory_working_set_bytes in the current working set) (Memory usage in a working set = Active anonymous page and cache, and file-baked page ≤ container_memory_usage_bytes)	MB	≥ 0	N/A	N/A	N/A
Stor age	Disk Read Rate	ma_cont ainer_dis k_read_k ilobytes	Volume of data read from a disk per second	KB/s	≥ 0	N/A	N/A	N/A
	Disk Write Rate	ma_cont ainer_dis k_write_ kilobytes	Volume of data written into a disk per second	KB/s	≥ 0	N/A	N/A	N/A
GPU me mor y	Total GPU Memory	ma_cont ainer_gp u_mem_t otal_me gabytes	Total GPU memory of a training job	MB	> 0	N/A	N/A	N/A
	GPU Memory Usage	ma_cont ainer_gp u_mem_ util	Percentage of the used GPU memory to the total GPU memory	%	0%–100 %	N/A	N/A	N/A

Cate gory	Name	Metric	Description	Unit	Val ue Ran ge	Alar m Thres hold	Al ar m Se ve rit y	Solut ion
	Used GPU Memory	ma_cont ainer_gp u_mem_ used_me gabytes	GPU memory used by a measured object	MB	≥ 0	N/A	N/A	N/A
	Idle GPU Memory	ma_cont ainer_gp u_mem_f ree_meg abytes	Idle GPU memory of a measured object	MB	≥ 0	N/A	N/A	N/A
GPU	GPU usage	ma_cont ainer_gp u_util	GPU usage of a measured object	%	0%– 100 %	Raw data > 95% for two consecutive periods	Su gg est ion	Chec k whet her the servi ce resou rce usag e meet s the expe ctati on. If the servi ce is norm al, no actio n is requi red.

Cate gory	Name	Metric	Description	Unit	Val ue Ran ge	Alar m Thres hold	Al ar m Se ve rit y	Solut ion
	GPU Memory Bandwid th Usage	ma_cont ainer_gp u_mem_ copy_util	GPU memory bandwidth usage of a measured object For example, the maximum memory bandwidth of GP Vnt1 is 900 GB/s. If the current memory bandwidth is 450 GB/s, the memory bandwidth usage is 50%.	%	0%–100 %	N/A	N/A	N/A
	GPU Encoder Usage	ma_cont ainer_gp u_enc_util	GPU encoder usage of a measured object	%	%	N/A	N/A	N/A
	GPU Decoder Usage	ma_cont ainer_gp u_dec_util	GPU decoder usage of a measured object	%	%	N/A	N/A	N/A
	GPU Tempera ture	DCGM_F I_DEV_G PU_TEMP	GPU temperature	°C	Nat ural num ber	N/A	N/A	N/A
	GPU Power	DCGM_F I_DEV_P OWER_U SAGE	GPU power	Watt (W)	> 0	N/A	N/A	N/A
	GPU Memory Tempera ture	DCGM_F I_DEV_M EMORY_ TEMP	GPU memory temperature	°C	Nat ural num ber	N/A	N/A	N/A

Category	Name	Metric	Description	Unit	Value Range	Alarm Threshold	Alarm Severity	Solution
Network I/O	Downlink Rate	ma_container_network_receive_bytes	Inbound traffic rate of a measured object	Bytes/s	≥ 0	N/A	N/A	N/A
	Packet RX Rate	ma_container_network_receive_packets	Number of data packets received by a NIC per second	Packets/s	≥ 0	N/A	N/A	N/A
	Downlink Error Rate	ma_container_network_receive_error_packets	Number of error packets received by a NIC per second	Packets/s	≥ 0	Raw data > 1 for two consecutive periods	Critical	Packet loss on the network. Submit a service ticket and contact the O&M support to locate the fault.
	Uplink Rate	ma_container_network_transmit_bytes	Outbound traffic rate of a measured object	Bytes/s	≥ 0	N/A	N/A	N/A

Category	Name	Metric	Description	Unit	Value Range	Alarm Threshold	Alarm Severity	Solution
	Uplink Error Rate	ma_container_network_transmit_error_packets	Number of error packets sent by a NIC per second	Packets/s	≥ 0	Raw data > 1 for two consecutive periods	Critical	Packet loss on the network. Submit a service ticket and contact the O&M support to locate the fault.
	Packet TX Rate	ma_container_network_transmit_packets	Number of data packets sent by a NIC per second	Packets/s	≥ 0	N/A	N/A	N/A

Cate gory	Name	Metric	Description	Unit	Val ue Ran ge	Alar m Thres hold	Al ar m Se ve rit y	Solut ion
NPU	NPU Usage	ma_cont ainer_np u_util	NPU usage of a measured object (To be replaced by ma_container _npu_ai_core _util)	%	0%– 100 %	Raw data > 95% for two conse cutiv e perio ds	Su gg est ion	Chec k whet her the servi ce resou rce usag e meet s the expe ctati on. If the servi ce is norm al, no actio n is requi red.

Category	Name	Metric	Description	Unit	Value Range	Alarm Threshold	Alarm Severity	Solution
	NPU Memory Usage	ma_container_npu_memory_util	Percentage of the used NPU memory to the total NPU memory (To be replaced by ma_container_npu_ddr_memory_util for Snt3 series, and ma_container_npu_hbm_util for Snt9 series)	%	0%–100%	Raw data > 98% for two consecutive periods	Suggestion	Check whether the service resource usage meets the expectation. If the service is normal, no action is required.
	Used NPU Memory	ma_container_npu_memory_used_megabytes	NPU memory used by a measured object (To be replaced by ma_container_npu_ddr_memory_usage_bytes for Snt3 series, and ma_container_npu_hbm_usage_bytes for Snt9 series)	≥ 0	MB	N/A	N/A	N/A

Category	Name	Metric	Description	Unit	Value Range	Alarm Threshold	Alarm Severity	Solution
	Total NPU Memory	ma_container_npu_memory_total_megabytes	Total NPU memory of a measured object (To be replaced by ma_container_npu_ddr_memory_bytes for Snt3 series, and ma_container_npu_hbm_bytes for Snt9 series)	> 0	MB	N/A	N/A	N/A
	AI Processor Error Codes	ma_container_npu_ai_core_error_code	Error codes of Ascend AI processors	-	-	Raw data > 0 for three consecutive periods	Critical	Abnormal card. Submit a service ticket and contact the O&M support.

Category	Name	Metric	Description	Unit	Value Range	Alarm Threshold	Alarm Severity	Solution
	AI Processor Health Status	ma_container_npu_ai_core_health_status	Health status of Ascend AI processors	-	1: healthy 0: unhealthy	Raw data > 0 for two consecutive periods	Critical	Abnormal card. Submit a service ticket and contact the O&M support.
	AI Processor Power Consumption	ma_container_npu_ai_core_power_usage_watts	Power consumption of Ascend AI processors	Watt (W)	> 0	N/A	N/A	N/A
	AI Processor Temperature	ma_container_npu_ai_core_temperature_celsius	Temperature of Ascend AI processors	°C	Natural number	N/A	N/A	N/A

Cate gory	Name	Metric	Description	Unit	Val ue Ran ge	Alar m Thres hold	Al ar m Se ve rit y	Solut ion
	AI Core Usage	ma_cont ainer_np u_ai_core _util	AI core usage of Ascend AI processors	%	0%–100 %	Raw data > 95% for two consecutive periods	Su gg est ion	Chec k whet her the servi ce resou rce usag e meet s the expe ctati on. If the servi ce is norm al, no actio n is requi red.
	Overall NPU Usage	ma_cont ainer_np u_genera l_util	NPU usage of Ascend AI processors (supported by driver version 24.1.RC2 and later)	%	0%–100 %	N/A	N/ A	N/A
	AI Core Clock Frequency	ma_cont ainer_np u_ai_core _frequen cy_hertz	AI core clock frequency of Ascend AI processors	Hertz (Hz)	> 0	N/A	N/ A	N/A

Category	Name	Metric	Description	Unit	Value Range	Alarm Threshold	Alarm Severity	Solution
	AI Processor Voltage	ma_container_npu_ai_core_voltage_volts	Voltage of Ascend AI processors	Volt (V)	Natural number	N/A	N/A	N/A
	AI Processor DDR Memory	ma_container_npu_ddr_memory_bytes	Total DDR memory capacity of Ascend AI processors	Byte	> 0	N/A	N/A	N/A
	AI Processor DDR Usage	ma_container_npu_ddr_memory_usage_bytes	DDR memory usage of Ascend AI processors	Byte	> 0	N/A	N/A	N/A
	AI Processor DDR Memory Utilization	ma_container_npu_ddr_memory_util	DDR memory utilization of Ascend AI processors Invalid metric for Snt9C.	%	0%–100%	Raw data > 95% for two consecutive periods	Suggestion	Check whether the service resource usage meets the expectation. If the service is normal, no action is required.

Category	Name	Metric	Description	Unit	Value Range	Alarm Threshold	Alarm Severity	Solution
	AI Processor HBM Memory	ma_container_npu_hbm_bytes	Total HBM memory of Ascend AI processors (dedicated for Snt9 processors)	Byte	> 0	N/A	N/A	N/A
	AI Processor HBM Memory Usage	ma_container_npu_hbm_usage_bytes	HBM memory usage of Ascend AI processors (dedicated for Snt9 processors)	Byte	> 0	N/A	N/A	N/A
	AI Processor HBM Memory Utilization	ma_container_npu_hbm_util	HBM memory utilization of Ascend AI processors (dedicated for Snt9 processors)	%	0%–100%	Raw data > 95% for two consecutive periods	Suggestion	Check whether the service resource usage meets the expectation. If the service is normal, no action is required.

Cate gory	Name	Metric	Description	Unit	Val ue Ran ge	Alar m Thres hold	Al ar m Se ve rit y	Solut ion
	AI Processo r HBM Memory Bandwid th Utilizatio n	ma_cont ainer_np u_hbm_b andwidt h_util	HBM memory bandwidth utilization of Ascend AI processors (dedicated for Snt9 processors)	%	0%– 100 %	Raw data > 95% for two conse cutiv e perio ds	Su gg est ion	Chec k whet her the servi ce resou rce usag e meet s the expe ctati on. If the servi ce is norm al, no actio n is requi red.
	AI Processo r HBM Memory Clock Frequenc y	ma_cont ainer_np u_hbm_f requency _hertz	HBM memory clock frequency of Ascend AI processors (dedicated for Snt9 processors)	Hertz (Hz)	> 0	N/A	N/ A	N/A
	AI Processo r HBM Memory Tempera ture	ma_cont ainer_np u_hbm_t emperat ure_celsi us	HBM memory temperature of Ascend AI processors (dedicated for Snt9 processors)	°C	Nat ural num ber	N/A	N/ A	N/A

Cate gory	Name	Metric	Description	Unit	Val ue Ran ge	Alar m Thres hold	Al ar m Se ve rit y	Solut ion
	AI CPU Utilization	ma_cont ainer_np u_ai_cpu _util	AI CPU utilization of Ascend AI processors	%	0%–100 %	N/A	N/A	N/A
	AI Processor Control CPU Utilization	ma_cont ainer_np u_ctrl_cp u_util	Control CPU utilization of Ascend AI processors	%	0%–100 %	N/A	N/A	N/A
	AI Processor Control CPU Frequency	ma_node _npu_ctrl _cpu_fre quency_ hertz	Control CPU frequency of Ascend AI processors	Hertz (Hz)	> 0 Syst em mod e (ded icat ed reso urce pool user mod e)	N/A	N/A	N/A

Cate gory	Name	Metric	Description	Unit	Val ue Ran ge	Alar m Thres hold	Al ar m Se ve rit y	Solut ion
	AI Vector Core Usage	ma_cont ainer_np u_vector _core_uti l	AI vector core usage of Ascend AI processors	%	0%– 100 %	Raw data > 95% for two conse cutiv e perio ds	Su gg est ion	Chec k whet her the servi ce resou rce usag e meet s the expe ctati on. If the servi ce is norm al, no actio n is requi red.
	Successf ul NPU Operator Retrans missions	ma_cont ainer_np u_operat or_retry_ success_c nt	Number of successful NPU operator retransmission s (supported by A3 Ascend HDK 24.1.RC3.3 or later)	Num ber	≥ 0			

Category	Name	Metric	Description	Unit	Value Range	Alarm Threshold	Alarm Severity	Solution
	Failed NPU Operator Retransmissions	ma_container_npu_operator_retry_fail_cnt	Number of failed NPU operator retransmissions (supported by A3 Ascend HDK 24.1.RC3.3 or later)	Number	≥ 0			
	Number of Times that an NPU Is Used for Communication	ma_container_npu_borrow_comm_cnt	Number of times that an NPU is used for communication. The more the NPU is used, the lower the transmission efficiency is. (supported by A3 Ascend HDK 24.1.RC3.3 or later)	Number	≥ 0			
NPU RoCE Network	NPU RoCE Network Uplink Rate	ma_container_npu_roce_tx_rate_bytes_per_second	Uplink rate of the NPU network module used by the container	Bytes/s	≥ 0	N/A	N/A	N/A
	NPU RoCE Network Downlink Rate	ma_container_npu_roce_rx_rate_bytes_per_second	Downlink rate of the NPU network module used by the container	Bytes/s	≥ 0	N/A	N/A	N/A

Category	Name	Metric	Description	Unit	Value Range	Alarm Threshold	Alarm Severity	Solution
Notebook service metrics	Notebook Cache Directory Size	ma_container_notebook_cache_dir_size_bytes	A high-speed local disk is attached to the /cache directory for GPU and NPU notebook instances. This metric indicates the total size of the directory.	Bytes	≥ 0	N/A	N/A	N/A
	Notebook Cache Directory Utilization	ma_container_notebook_cache_dir_util	A high-speed local disk is attached to the /cache directory for GPU and NPU notebook instances. This metric indicates the utilization of the directory.	%	0%–100%	Raw data > 90% for two consecutive periods	Major	If the disk usage is too high, the notebook instance will be restarted.

Node-level Metrics

Table 10-2 Node metrics (collected only in dedicated resource pools)

Cat ego ry	Name	Metric	Description	Unit	Val ue Ran ge	Alarm Thresh old	Al ar m Se ve rit y	Soluti on
CPU	Total CPU Cores	ma_nod e_cpu_li mit_core	Total number of CPU cores that have been requested for a measured object	Cores	≥ 1	N/A	N/ A	N/A
	Used CPU Cores	ma_nod e_cpu_us ed_core	Number of CPU cores used by a measured object	Cores	≥ 0	N/A	N/ A	N/A
	CPU Usage	ma_nod e_cpu_ut il	CPU usage of a measured object	%	0% – 100 %	Raw data > 95% for two consecutive periods	M ajo r	Check whether the service resource usage meets the expectation . If the service is normal, no action is required.
	CPU I/O Wait Time	ma_nod e_cpu_io wait_cou nter	Disk I/O wait time accumulated since system startup	jiffies	≥ 0	N/A	N/ A	N/A

Cat ego ry	Name	Metric	Description	Unit	Val ue Ran ge	Alarm Thresh old	Al arm Se ve rit y	Soluti on
Me mor y	Physica l Mem ory Usage	ma_nod e_memo ry_util	Percentage of the used physical memory to the total physical memory	%	0% – 100 %	Raw data > 95% for two consecu tive periods	M ajo r	Check wheth er the servic e resour ce usage meets the expectation . If the servic e is norm al, no action is requir ed.
	Total Physica l Mem ory	ma_nod e_memo ry_total_ megabyt es	Total physical memory that has been applied for a measured object	MB	≥ 0	N/A	N/ A	N/A
Net wor k I/O	Downli nk Rate (BPS)	ma_nod e_netwo rk_receiv e_rate_b ytes_sec onds	Inbound traffic rate of a measured object	Bytes /s	≥ 0	N/A	N/ A	N/A
	Uplink Rate (BPS)	ma_nod e_netwo rk_trans mit_rate_ bytes_s econds	Outbound traffic rate of a measured object	Bytes /s	≥ 0	N/A	N/ A	N/A

Cat ego ry	Name	Metric	Description	Unit	Val ue Ran ge	Alarm Thresh old	Al arm Se ve rit y	Soluti on
Stor age	Disk Read Rate	ma_nod e_disk_r ead_rate _kilobyte s_second s	Volume of data read from a disk per second (Only data disks used by containers are collected.)	KB/s	≥ 0	N/A	N/A	N/A
	Disk Write Rate	ma_nod e_disk_w rite_rate _kilobyte s_second s	Volume of data written into a disk per second (Only data disks used by containers are collected.)	KB/s	≥ 0	N/A	N/A	N/A
	Total Cache	ma_nod e_cache_ space_ca pacity_m egabytes	Total cache of the Kubernetes space	MB	≥ 0	N/A	N/A	N/A
	Used Cache	ma_nod e_cache_ space_us ed_capa city_meg abytes	Used cache of the Kubernetes space	MB	≥ 0	N/A	N/A	N/A

Cat ego ry	Name	Metric	Description	Unit	Val ue Ran ge	Alarm Thresh old	Al ar m Se ve rit y	Soluti on
	Cache Usage	ma_node_cache_space_used_percent	Cache usage of the Kubernetes space	%	≥ 0	Raw data > 90% for two consecutive periods	Critical	Check the disk in a timely manner to avoid affecting services. Clear invalid data on compute nodes.
	Total Container Space	ma_node_container_space_capacity_megabytes	Total container space	MB	≥ 0	N/A	N/A	N/A
	Used Container Space	ma_node_container_space_used_capacity_megabytes	Used container space	MB	≥ 0	N/A	N/A	N/A

Cat ego ry	Name	Metric	Description	Unit	Val ue Ran ge	Alarm Thresh old	Al ar m Se ve rit y	Soluti on
	Contai ner Space Usage	ma_nod e_contai ner_spac e_used_ percent	Space usage of a container	%	≥ 0	Raw data > 90% for two consecu tive periods	Cri tical	Check the disk in a timely mann er to avoid affect ing servic es. Clear invali d data on comp ute nodes .
	Disk Inform ation	ma_nod e_disk_in fo	Basic disk information	-	≥ 0	N/A	N/ A	N/A
	Total Reads	ma_nod e_disk_r eads_co mpleted _total	Total number of successful reads	-	≥ 0	N/A	N/ A	N/A
	Merge d Reads	ma_nod e_disk_r eads_me rged_tot al	Number of merged reads	-	≥ 0	N/A	N/ A	N/A
	Bytes Read	ma_nod e_disk_r ead_byte s_total	Total number of bytes that are successfully read	Bytes	≥ 0	N/A	N/ A	N/A

Cat ego ry	Name	Metric	Description	Unit	Val ue Ran ge	Alarm Thresh old	Al arm Se ve rit y	Soluti on
	Read Time Spent	ma_nod e_disk_r ead_tim e_secon ds_total	Time spent on all reads	Secon ds	≥ 0	N/A	N/ A	N/A
	Total Writes	ma_nod e_disk_w rites_co mpleted _total	Total number of successful writes	-	≥ 0	N/A	N/ A	N/A
	Merge d Writes	ma_nod e_disk_w rites_me rged_tot al	Number of merged writes	-	≥ 0	N/A	N/ A	N/A
	Writte n Bytes	ma_nod e_disk_w ritten_by tes_total	Total number of bytes that are successfully written	Bytes	≥ 0	N/A	N/ A	N/A
	Write Time Spent	ma_nod e_disk_w rite_time _seconds _total	Time spent on all write operations	Secon ds	≥ 0	N/A	N/ A	N/A
	Ongoin g I/Os	ma_nod e_disk_io _now	Number of ongoing I/Os	-	≥ 0	N/A	N/ A	N/A
	I/O Executi on Durati on	ma_nod e_disk_io _time_se conds_to tal	Time spent on executing I/Os	Secon ds	≥ 0	N/A	N/ A	N/A
	I/O Executi on Weight ed Time	ma_nod e_disk_io _time_w eighted_ seconds_ tota	Weighted time spent on executing I/Os	Secon ds	≥ 0	N/A	N/ A	N/A

Cat ego ry	Name	Metric	Description	Unit	Val ue Ran ge	Alarm Thresh old	Al arm Se ve rit y	Soluti on
GPU	GPU Usage	ma_node_gpu_util	GPU usage of a measured object	%	0% – 100%	N/A	N/A	N/A
	Total GPU Memory	ma_node_gpu_mem_total_megabytes	Total GPU memory of a measured object	MB	> 0	N/A	N/A	N/A
	GPU Memory Usage	ma_node_gpu_mem_util	Percentage of the used GPU memory to the total GPU memory	%	0% – 100%	Raw data > 97% for two consecutive periods	Suggestion	Check whether the service resource usage meets the expectation. If the service is normal, no action is required.
	Used GPU Memory	ma_node_gpu_mem_used_megabytes	GPU memory used by a measured object	MB	≥ 0	N/A	N/A	N/A
	Idle GPU Memory	ma_node_gpu_mem_free_megabytes	Idle GPU memory of a measured object	MB	> 0	N/A	N/A	N/A

Cat ego ry	Name	Metric	Description	Unit	Val ue Ran ge	Alarm Thresh old	Al arm Se ve rit y	Soluti on
	Tasks on a Shared GPU	node_gpu_share_job_count	Number of tasks running on a shared GPU	Num ber	≥ 0	N/A	N/A	N/A
	GPU Temperature	DCGM_FI_DEV_GPU_TEMP	GPU temperature	°C	Nat ural nu mb er	N/A	N/A	N/A
	GPU Power	DCGM_FI_DEV_POWER_USAGE	GPU power	Watt (W)	> 0	N/A	N/A	N/A
	GPU Memory Temperature	DCGM_FI_DEV_MEMORY_TEMP	GPU memory temperature	°C	Nat ural nu mb er	N/A	N/A	N/A
NPU	NPU Usage	ma_node_npu_util	NPU usage of a measured object (To be replaced by ma_node_npu_ai_core_util)	%	0% – 100 %	N/A	N/A	N/A

Cat ego ry	Name	Metric	Description	Unit	Val ue Ran ge	Alarm Thresh old	Al ar m Se ve rit y	Soluti on
	NPU Memor y Usage	ma_nod e_npu_ memory _util	Percentage of the used NPU memory to the total NPU memory (To be replaced by ma_node_np u_ddr_memo ry_util for Snt3 series, and ma_node_np u_hbm_util for Snt9 series)	%	0% – 100 %	Raw data > 97% for two consecu tive periods	Su gg est ion	Check wheth er the servic e resour ce usage meets the expectation . If the servic e is norm al, no action is requir ed.
	Used NPU Memor y	ma_nod e_npu_ memory _used_m egabytes	NPU memory used by a measured object (To be replaced by ma_node_np u_ddr_memo ry_usage_byt es for Snt3 series, and ma_node_np u_hbm_usag e_bytes for Snt9 series)	≥ 0	MB	N/A	N/ A	N/A

Cat ego ry	Name	Metric	Description	Unit	Val ue Ran ge	Alarm Thresh old	Al arm Se ve rit y	Soluti on
	Total NPU Memory	ma_node_npu_memory_total_megabytes	Total NPU memory of a measured object (To be replaced by ma_node_npu_ddr_memory_bytes for Snt3 series, and ma_node_npu_hbm_bytes for Snt9 series)	> 0	MB	N/A	N/A	N/A
	AI Processor Error Codes	ma_node_npu_ai_core_error_code	Error codes of Ascend AI processors	-	-	N/A	N/A	N/A
	AI Processor Health Status	ma_node_npu_ai_core_health_status	Health status of Ascend AI processors	-	• 1: healthy • 0: unhealthy	The value is 0 for two consecutive periods.	Critical	Submit a service ticket.

Cat ego ry	Name	Metric	Description	Unit	Val ue Ran ge	Alarm Thresh old	Al arm Se ve rit y	Soluti on
	AI Process or Power Consumption	ma_node_npu_ai_core_power_usage_watts	Power consumption of Ascend AI processors	Watt (W)	> 0	N/A	N/A	N/A
	AI Process or Temperature	ma_node_npu_ai_core_temperature_celsius	Temperature of Ascend AI processors	°C	Natural number	N/A	N/A	N/A
	AI Core Usage	ma_node_npu_ai_core_util	AI core usage of Ascend AI processors	%	0% – 100%	N/A	N/A	N/A
	Overall NPU Usage	ma_node_npu_general_util	NPU usage of Ascend AI processors (supported by driver version 24.1.RC2 and later)	%	0% – 100%	N/A	N/A	N/A
	AI Core Clock Frequency	ma_node_npu_ai_core_frequency_hertz	AI core clock frequency of Ascend AI processors	Hertz (Hz)	> 0	N/A	N/A	N/A
	AI Process or Voltage	ma_node_npu_ai_core_voltage_volts	Voltage of Ascend AI processors	Volt (V)	Natural number	N/A	N/A	N/A

Cat ego ry	Name	Metric	Description	Unit	Val ue Ran ge	Alarm Thresh old	Al arm Se ve rit y	Soluti on
	AI Process or DDR Memor y	ma_nod e_npu_d dr_mem ory_byte s	Total DDR memory capacity of Ascend AI processors Invalid metric for Snt9C.	Byte	> 0	N/A	N/ A	N/A
	AI Process or DDR Usage	ma_nod e_npu_d dr_mem ory_usag e_bytes	DDR memory usage of Ascend AI processors	Byte	> 0	N/A	N/ A	N/A
	AI Process or DDR Memor y Utilizat ion	ma_nod e_npu_d dr_mem ory_util	DDR memory utilization of Ascend AI processors	%	0% – 100 %	Raw data > 90% for two consecu tive periods	Su gg est ion	Check wheth er the servic e resour ce usage meets the expecta tion . If the servic e is norma l, no action is requir ed.
	AI Process or HBM Memor y	ma_nod e_npu_h bm_byte s	Total HBM memory of Ascend AI processors (dedicated for Snt9 processors)	Byte	> 0	N/A	N/ A	N/A

Cat ego ry	Name	Metric	Description	Unit	Val ue Ran ge	Alarm Thresh old	Al arm Se ve rit y	Soluti on
	AI Process or HBM Memor y Usage	ma_node_npu_hbm_usage_bytes	HBM memory usage of Ascend AI processors (dedicated for Snt9 processors)	Byte	> 0	N/A	N/A	N/A
	AI Process or HBM Memor y Utilizat ion	ma_node_npu_hbm_util	HBM memory utilization of Ascend AI processors (dedicated for Snt9 processors)	%	0% – 100%	Raw data > 97% for two consecutive periods	Su gg est ion	Check whether the service resource usage meets the expectation. If the service is normal, no action is required.
	AI Process or HBM Memor y Bandwi dth Utilizat ion	ma_node_npu_hbm_bandwidth_util	HBM memory bandwidth utilization of Ascend AI processors (dedicated for Snt9 processors)	%	0% – 100%	N/A	N/A	N/A

Cat ego ry	Name	Metric	Description	Unit	Val ue Ran ge	Alarm Thresh old	Al ar m Se ve rit y	Soluti on
	AI Process or HBM Memory Clock Freque ncy	ma_nod e_npu_h bm_freq uency_h ertz	HBM memory clock frequency of Ascend AI processors (dedicated for Snt9 processors)	Hertz (Hz)	> 0	N/A	N/ A	N/A
	AI Process or HBM Memory Tempe rature	ma_nod e_npu_h bm_tem perature _celsius	HBM memory temperature of Ascend AI processors (dedicated for Snt9 processors)	°C	Nat ural nu mb er	N/A	N/ A	N/A
	AI CPU Utilizat ion	ma_nod e_npu_ai _cpu_util	AI CPU utilization of Ascend AI processors	%	0% – 100 %	N/A	N/ A	N/A
	AI Process or Control CPU Utilizat ion	ma_nod e_npu_ct rl_cpu_u til	Control CPU utilization of Ascend AI processors	%	0% – 100 %	N/A	N/ A	N/A

Cat ego ry	Name	Metric	Description	Unit	Val ue Ran ge	Alarm Thresh old	Al ar m Se ve rit y	Soluti on
	AI Process or Control CPU Freque ncy	ma_nod e_npu_ct rl_cpu_fr equency _hertz	Control CPU frequency of Ascend AI processors	Hertz (Hz)	> 0 Syst em mo de (av aila ble for ded icat ed reso urc e poo l use rs)	N/A	N/ A	N/A
	HBM ECC Detecti on Switch	ma_nod e_npu_h bm_ecc_ enable	0 indicates that ECC detection is disabled. 1 indicates that ECC detection is enabled.	-	• 1 : e n a b l e d • 0 : d i s a b l e d	N/A	N/ A	N/A

Cat ego ry	Name	Metric	Description	Unit	Val ue Ran ge	Alarm Thresh old	Al ar m Se ve rit y	Soluti on
	Current HBM Single-bit Errors	ma_node_npu_hbm_single_bit_error_total	Current number of HBM single-bit errors	Number	≥ 0	N/A	N/A	N/A
	Current HBM Multi-bit Errors	ma_node_npu_hbm_double_bit_error_total	Current number of HBM multi-bit errors	Number	≥ 0	N/A	N/A	N/A
	Total Single-bit Errors in the HBM Life Cycle	ma_node_npu_hbm_total_single_bit_error_total	Total number of single-bit errors in the HBM life cycle	Number	≥ 0	N/A	N/A	N/A
	Total Multi-bit Errors in the HBM Life Cycle	ma_node_npu_hbm_total_double_bit_error_total	Total number of multi-bit errors in the HBM life cycle	Number	≥ 0	N/A	N/A	N/A
	Isolated NPU Memory Pages with HBM Single-bit Errors	ma_node_npu_hbm_single_bit_isolated_pages_total	Number of isolated NPU memory pages with HBM single-bit errors	Number	≥ 0	N/A	N/A	N/A

Cat ego ry	Name	Metric	Description	Unit	Val ue Ran ge	Alarm Thresh old	Al ar m Se ve rit y	Soluti on
	Isolate d NPU Memor y Pages with HBM Multi- bit Errors	ma_nod e_npu_h bm_dou ble_bit_i solated_ pages_to tal	Number of isolated NPU memory pages with HBM multi- bit errors Note: If there are more than 64 pages, change the NPU.	Num ber	≥ 0	Raw data $\geq$ 64 for two consecu tive periods	Cri tical	If there are more than 64 pages , submi t a servic e ticket, and switc h the NPU server .
	AI Vector Core Usage	ma_nod e_npu_v ector_co re_util	AI vector core usage of Ascend AI processors	%	0% – 100 %	N/A	N/ A	N/A
	NPU Macro Packet Retran smis sions	ma_nod e_npu_ macro_r etry_cnt	Number of NPU Macro packet retransmissio ns within a detection period (10 seconds) (supported by A3 24.1.RC2 or later)	Num ber	≥ 0			

Cat ego ry	Name	Metric	Description	Unit	Val ue Ran ge	Alarm Thresh old	Al arm Se ve rit y	Soluti on
	Packets Received by NPU Macro	ma_node_npu_macro_rx_cnt	Number of packets received by NPU Macro within a detection period (10 seconds)	Number	≥ 0			
	Invalid Packets Received by NPU Macro	ma_node_npu_macro_crc_error_cnt	Number of invalid CRC packets received by NPU Macro within a detection period (10 seconds)	Number	≥ 0			
	NPU Macro BER	ma_node_npu_macro_crc_error_rate	The percentage of invalid CRC packets received by NPU Macro within a detection period	%	0% – 100%			
	Successful NPU Operator Retransmissions	ma_node_npu_operator_retry_success_cnt	Number of successful NPU operator retransmissions (supported by A3 Ascend HDK 24.1.RC3.3 or later)	Number	≥ 0			

Cat ego ry	Name	Metric	Description	Unit	Val ue Ran ge	Alarm Thresh old	Al ar m Se ve rit y	Soluti on
	Failed NPU Operator Retransmissions	ma_node_npu_operator_retry_fail_cnt	Number of failed NPU operator retransmissions (supported by A3 Ascend HDK 24.1.RC3.3 or later)	Num ber	≥ 0			
	Number of Times that an NPU Is Used for Communication	ma_node_npu_borrow_comms_cnt	Number of times that an NPU is used for communication. The more the NPU is used, the lower the transmission efficiency is (supported by A3 Ascend HDK 24.1.RC3.3 or later).	Num ber	≥ 0			
	Number of Faulty 1520 Switches	ma_node_switch_fault_count	Number of faulty 1520 switches. This is a metric dedicated to 910C. The severity level can be 0, 1, 2, or 3.	Num ber	≥ 0	Raw data ≥ 0 for four consecutive periods	M ajo r	Subm it a servic e ticket.

Cat ego ry	Name	Metric	Description	Unit	Val ue Ran ge	Alarm Thresh old	Al ar m Se ve rit y	Soluti on
NPU HCC S Link s (A3 Spec ifica tion s)	Availab le NPU L/Cs	ma_npu_hccs_avail_credit	Number of available L/Cs, which is used to measure the capability of continuing to receive data (supported by driver version 25.1.RC1 or later).	Num ber	≥ 0			
	Total TX Bandwidth of HCCS Links	ma_node_npu_hccs_total_txbw_per_second	Total bandwidth of data sent by the NPU (supported by driver version 24.1.RC3.5 or later)	GB/s	≥ 0			
	Total RX Bandwidth of HCCS Links	ma_node_npu_hccs_total_rxbw_per_second	Total bandwidth of data received by the NPU (supported by driver version 24.1.RC3.5 or later)	GB/s	≥ 0			
	HCCS Link TX Bandwidth Details	ma_node_npu_hccs_txbw_per_second	Bandwidth sent by the NPU and each switch chip (supported by driver version 24.1.RC3.5 or later)	GB/s	≥ 0			

Cat ego ry	Name	Metric	Description	Unit	Val ue Ran ge	Alarm Thresh old	Al ar m Se ve rit y	Soluti on
	HCCS Link TX Bandwi dth Details	ma_nod e_npu_h ccs_rxbw _per_sec ond	Bandwidth received by the NPU and each switch chip (supported by driver version 24.1.RC3.5 or later)	GB/s	≥ 0			
NPU RoCE net wor k	NPU RoCE Netwo rk Uplink Rate	ma_nod e_npu_r oce_tx_r ate_byte s_per_se cond	NPU RoCE network uplink rate	Bytes /s	≥ 0	N/A	N/ A	N/A
	NPU RoCE Netwo rk Downli nk Rate	ma_nod e_npu_r oce_rx_r ate_byte s_per_se cond	NPU RoCE network downlink rate	Bytes /s	≥ 0	N/A	N/ A	N/A
	MAC Uplink Pause Frames	ma_nod e_npu_r oce_mac _tx_paus e_packet s_total	Total number of pause frame packets sent by NPU RoCE network MAC	Num ber	≥ 0	N/A	N/ A	N/A
	MAC Downli nk Pause Frames	ma_nod e_npu_r oce_mac _rx_paus e_packet s_total	Total number of pause frame packets received by NPU RoCE network MAC	Num ber	≥ 0	N/A	N/ A	N/A

Cat ego ry	Name	Metric	Description	Unit	Val ue Ran ge	Alarm Thresh old	Al arm Se ve rit y	Soluti on
	MAC Uplink PFC Frames	ma_node_npu_roce_mac_tx_pfc_packets_total	Total number of PFC frame packets sent by NPU RoCE network MAC	Num ber	≥ 0	$\text{delta}(\text{ma\_node\_npu\_roce\_mac\_tx\_pause\_packets\_total}[1\text{m}]) > 0$	M ajo r	Subm it a servic e ticket.
	MAC Downlink PFC Frames	ma_node_npu_roce_mac_rx_pfc_packets_total	Total number of PFC frame packets received by NPU RoCE network MAC	Num ber	≥ 0	$\text{delta}(\text{ma\_node\_npu\_roce\_mac\_rx\_pause\_packets\_total}[1\text{m}]) > 0$	M ajo r	Subm it a servic e ticket.
	MAC Uplink Bad Packets	ma_node_npu_roce_mac_tx_bad_packets_total	Total number of bad packets sent by NPU RoCE network MAC	Num ber	≥ 0	$\text{delta}(\text{ma\_node\_npu\_roce\_mac\_tx\_pfc\_packets\_total}[1\text{m}]) > 0$	M ajo r	Subm it a servic e ticket.
	MAC Downlink Bad Packets	ma_node_npu_roce_mac_rx_bad_packets_total	Total number of bad packets received by NPU RoCE network MAC	Num ber	≥ 0	$\text{delta}(\text{ma\_node\_npu\_roce\_mac\_rx\_pfc\_packets\_total}[1\text{m}]) > 0$	M ajo r	Subm it a servic e ticket.
	RoCE Uplink Bad Packets	ma_node_npu_roce_tx_err_packets_total	Total number of bad packets sent by NPU RoCE	Num ber	≥ 0	$\text{delta}(\text{ma\_node\_npu\_roce\_mac\_tx\_bad\_packets\_total}[1\text{m}]) > 0$	M ajo r	Subm it a servic e ticket.

Cat ego ry	Name	Metric	Description	Unit	Val ue Ran ge	Alarm Thresh old	Al ar m Se ve rit y	Soluti on
	RoCE Downlink Bad Packets	ma_node_npu_roce_rx_err_packets_total	Total number of bad packets received by NPU RoCE	Number	≥ 0	$\text{delta}(\text{ma\_node\_npu\_roce\_mac\_rx\_bad\_packets\_total}[1\text{m}]) > 0$	Major	Submit a service ticket.
	RoCE Uplink Packets	ma_node_npu_roce_tx_all_packets_total	Total number of packets sent by NPU RoCE	Number	≥ 0	$\text{delta}(\text{ma\_node\_npu\_roce\_tx\_err\_packets\_total}[1\text{m}]) > 0$	Major	Submit a service ticket.
	RoCE Downlink Packets	ma_node_npu_roce_rx_all_packets_total	Total number of packets received by NPU RoCE	Number	≥ 0	$\text{delta}(\text{ma\_node\_npu\_roce\_rx\_err\_packets\_total}[1\text{m}]) > 0$	Major	Submit a service ticket.
	Number of RoCE Retry Packets	ma_node_npu_roce_new_pkt_rty_total	Number of RoCE Retry Packets	Number	≥ 0	N/A	N/A	N/A

Cat ego ry	Name	Metric	Description	Unit	Val ue Ran ge	Alarm Thresh old	Al ar m Se ve rit y	Soluti on
	Numbe r of PSN Packet s Receiv ed by RoCE > Expect ed PSN Packet s, or Duplic ate PSN Packet s	ma_nod e_npu_r oce_out_ of_order _total	Number of PSN Packets Received by RoCE > Expected PSN Packets, or Duplicate PSN Packets	Num ber	≥ 0	N/A	N/ A	N/A
	Numbe r of CNP Packet s Receiv ed by RoCE	ma_nod e_npu_r oce_rx_c np_pack ets_total	Number of CNP Packets Received by RoCE	Num ber	≥ 0	N/A	N/ A	N/A
	Numbe r of CNP Packet s Sent by RoCE	ma_nod e_npu_r oce_tx_c np_pack ets_total	Number of CNP Packets Sent by RoCE	Num ber	≥ 0	N/A	N/ A	N/A

Cat ego ry	Name	Metric	Description	Unit	Val ue Ran ge	Alarm Thresh old	Al arm Se ve rit y	Soluti on
NPU Opti cal Mod ule (Thi s met ric is avai lable for Snt9 B/C air- cool ed net wor king .)	Optical Modul e Tempe rature	ma_nod e_npu_o ptical_te mperatu re	Optical module temperature	°C	≥ 0	N/A	N/ A	N/A
	Optical Modul e Power Volutag e	ma_nod e_npu_o ptical_vc c	Power voltage of the optical module	Milliv olt (mV)	≥ 0	N/A	N/ A	N/A
	Optical Modul e Transm it Power 0	ma_nod e_npu_o ptical_tx _power0	Transmit power 0 of the optical module	Milli watt (mW)	≥ 0	N/A	N/ A	N/A
	Optical Modul e Transm it Power 1	ma_nod e_npu_o ptical_tx _power1	Transmit power 1 of the optical module	Milli watt (mW)	≥ 0	N/A	N/ A	N/A
	Optical Modul e Transm it Power 2	ma_nod e_npu_o ptical_tx _power2	Transmit power 2 of the optical module	Milli watt (mW)	≥ 0	N/A	N/ A	N/A
	Optical Modul e Transm it Power 3	ma_nod e_npu_o ptical_tx _power3	Transmit power 3 of the optical module	Milli watt (mW)	≥ 0	N/A	N/ A	N/A

Cat ego ry	Name	Metric	Description	Unit	Val ue Ran ge	Alarm Thresh old	Al ar m Se ve rit y	Soluti on
	Optical Modul e Receiv e Power 0	ma_nod e_npu_o ptical_rx _power0	Receive power 0 of the optical module	Milli watt (mW)	≥ 0	N/A	N/ A	N/A
	Optical Modul e Receiv e Power 1	ma_nod e_npu_o ptical_rx _power1	Receive power 1 of the optical module	Milli watt (mW)	≥ 0	N/A	N/ A	N/A
	Optical Modul e Receiv e Power 2	ma_nod e_npu_o ptical_rx _power2	Receive power 2 of the optical module	Milli watt (mW)	≥ 0	N/A	N/ A	N/A
	Optical Modul e Receiv e Power 3	ma_nod e_npu_o ptical_rx _power3	Receive power 3 of the optical module	Milli watt (mW)	≥ 0	N/A	N/ A	N/A
Infin iBan d or RoC E net wor k	Total Amoun t of Data Receiv ed by a NIC	ma_nod e_infinib and_port _receive d_data_ bytes_to tal	The total number of data octets, divided by 4, (counting in double words, 32 bits), received on all VLS from the port.	(coun ting in doubl e words , 32 bits)	≥ 0	N/A	N/ A	N/A

Cat ego ry	Name	Metric	Description	Unit	Val ue Ran ge	Alarm Thresh old	Al ar m Se ve rit y	Soluti on
	Total Amoun t of Data Sent by a NIC	ma_nod e_infinib and_port _transmi tted_dat a_bytes_ total	The total number of data octets, divided by 4, (counting in double words, 32 bits), transmitted on all VLS from the port.	(coun ting in doubl e words , 32 bits)	≥ 0	N/A	N/ A	N/A

Cat ego ry	Name	Metric	Description	Unit	Val ue Ran ge	Alarm Thresh old	Al ar m Se ve rit y	Soluti on
NFS mou ntin g stat us	NFS Getattr Conges tion Time	ma_nod e_mount stats_get attr_bac klog_wai t	Getattr is an NFS operation that retrieves the attributes of a file or directory, such as size, permissions, owner, etc. Backlog wait is the time that the NFS requests have to wait in the backlog queue before being sent to the NFS server. It indicates the congestion on the NFS client side. A high backlog wait can cause poor NFS performance and slow system response times.	ms	≥ 0	N/A	N/ A	N/A

Cat ego ry	Name	Metric	Description	Unit	Val ue Ran ge	Alarm Thresh old	Al ar m Se ve rit y	Soluti on
	NFS Getattr Round Trip Time	ma_nod e_mount stats_get attr_rtt	Getattr is an NFS operation that retrieves the attributes of a file or directory, such as size, permissions, owner, etc. RTT stands for Round Trip Time and it is the time from when the kernel RPC client sends the RPC request to the time it receives the reply ³⁴ . RTT includes network transit time and server execution time. RTT is a good measurement for NFS latency. A high RTT can indicate network or server issues.	ms	≥ 0	N/A	N/ A	N/A

Cat ego ry	Name	Metric	Description	Unit	Val ue Ran ge	Alarm Thresh old	Al arm Se ve rit y	Soluti on
	NFS Access Conges tion Time	ma_nod e_mount stats_acc ess_back log_wait	Access is an NFS operation that checks the access permissions of a file or directory for a given user. Backlog wait is the time that the NFS requests have to wait in the backlog queue before being sent to the NFS server. It indicates the congestion on the NFS client side. A high backlog wait can cause poor NFS performance and slow system response times.	ms	≥ 0	N/A	N/ A	N/A

Cat ego ry	Name	Metric	Description	Unit	Val ue Ran ge	Alarm Thresh old	Al ar m Se ve rit y	Soluti on
	NFS Access Round Trip Time	ma_nod e_mount stats_acc ess_rtt	Access is an NFS operation that checks the access permissions of a file or directory for a given user. RTT stands for Round Trip Time and it is the time from when the kernel RPC client sends the RPC request to the time it receives the reply ³⁴ . RTT includes network transit time and server execution time. RTT is a good measurement for NFS latency. A high RTT can indicate network or server issues.	ms	≥ 0	N/A	N/ A	N/A

Cat ego ry	Name	Metric	Description	Unit	Val ue Ran ge	Alarm Thresh old	Al arm Se ve rit y	Soluti on
	NFS Lookup Conges tion Time	ma_nod e_mount stats_loo kup_bac klog_wai t	Lookup is an NFS operation that resolves a file name in a directory to a file handle. Backlog wait is the time that the NFS requests have to wait in the backlog queue before being sent to the NFS server. It indicates the congestion on the NFS client side. A high backlog wait can cause poor NFS performance and slow system response times.	ms	≥ 0	N/A	N/ A	N/A

Cat ego ry	Name	Metric	Description	Unit	Val ue Ran ge	Alarm Thresh old	Al ar m Se ve rit y	Soluti on
	NFS Lookup Round Trip Time	ma_nod e_mount stats_loo kup_rtt	Lookup is an NFS operation that resolves a file name in a directory to a file handle. RTT stands for Round Trip Time and it is the time from when the kernel RPC client sends the RPC request to the time it receives the reply ³⁴ . RTT includes network transit time and server execution time. RTT is a good measurement for NFS latency. A high RTT can indicate network or server issues.	ms	≥ 0	N/A	N/ A	N/A

Cat ego ry	Name	Metric	Description	Unit	Val ue Ran ge	Alarm Thresh old	Al ar m Se ve rit y	Soluti on
	NFS Read Conges tion Time	ma_nod e_mount stats_rea d_backlo g_wait	Read is an NFS operation that reads data from a file. Backlog wait is the time that the NFS requests have to wait in the backlog queue before being sent to the NFS server. It indicates the congestion on the NFS client side. A high backlog wait can cause poor NFS performance and slow system response times.	ms	≥ 0	N/A	N/ A	N/A

Cat ego ry	Name	Metric	Description	Unit	Val ue Ran ge	Alarm Thresh old	Al ar m Se ve rit y	Soluti on
	NFS Read Round Trip Time	ma_nod e_mount stats_rea d_rtt	Read is an NFS operation that reads data from a file. RTT stands for Round Trip Time and it is the time from when the kernel RPC client sends the RPC request to the time it receives the reply ³⁴ . RTT includes network transit time and server execution time. RTT is a good measurement for NFS latency. A high RTT can indicate network or server issues.	ms	≥ 0	N/A	N/ A	N/A

Cat ego ry	Name	Metric	Description	Unit	Val ue Ran ge	Alarm Thresh old	Al ar m Se ve rit y	Soluti on
	NFS Write Conges tion Time	ma_nod e_mount stats_wri te_backl og_wait	Write is an NFS operation that writes data to a file. Backlog wait is the time that the NFS requests have to wait in the backlog queue before being sent to the NFS server. It indicates the congestion on the NFS client side. A high backlog wait can cause poor NFS performance and slow system response times.	ms	≥ 0	N/A	N/ A	N/A

Cat ego ry	Name	Metric	Description	Unit	Val ue Ran ge	Alarm Thresh old	Al arm Se ve rit y	Soluti on
	NFS Write Round Trip Time	ma_nod e_mount stats_wri te_rtt	Write is an NFS operation that writes data to a file. RTT stands for Round Trip Time and it is the time from when the kernel RPC client sends the RPC request to the time it receives the reply ³⁴ . RTT includes network transit time and server execution time. RTT is a good measurement for NFS latency. A high RTT can indicate network or server issues.	ms	≥ 0	N/A	N/ A	N/A

Networking Metrics

Table 10-3 Diagnosis (InfiniBand, collected only in dedicated resource pools)

Category	Name	Metric	Description	Unit	Value Range
InfiniB and or RoCE network	PortXmitData	infiniband_port_xmit_data_total	The total number of data octets, divided by 4, (counting in double words, 32 bits), transmitted on all VLs from the port.	Total count	Natural number
	PortRcvData	infiniband_port_rcv_data_total	The total number of data octets, divided by 4, (counting in double words, 32 bits), received on all VLs from the port.	Total count	Natural number
	SymbolErrorCounter	infiniband_symbol_error_counter_total	Total number of minor link errors detected on one or more physical lanes.	Total count	Natural number
	LinkErrorRecoveryCounter	infiniband_link_error_recovery_counter_total	Total number of times the Port Training state machine has successfully completed the link error recovery process.	Total count	Natural number

Category	Name	Metric	Description	Unit	Value Range
	PortRcvErrors	infiniband_port_rcv_errors_total	Total number of packets containing errors that were received on the port including: Local physical errors (ICRC, VCRC, LPCRC, and all physical errors that cause entry into the BAD PACKET or BAD PACKET DISCARD states of the packet receiver state machine) Malformed data packet errors (LVer, length, VL) Malformed link packet errors (operand, length, VL) Packets discarded due to buffer overrun (overflow)	Total count	Natural number
	LocalLinkIntegrityErrors	infiniband_local_link_integrity_errors_total	This counter indicates the number of retries initiated by a link transfer layer receiver.	Total count	Natural number
	PortRcvRemotePhysicalErrors	infiniband_port_rcv_remote_physical_errors_total	Total number of packets marked with the EBP delimiter received on the port.	Total count	Natural number
	PortRcvSwitchRelayErrors	infiniband_port_rcv_switch_relay_errors_total	Total number of packets received on the port that were discarded when they could not be forwarded by the switch relay for the following reasons: DLID mapping VL mapping Looping (output port = input port)	Total count	Natural number

Category	Name	Metric	Description	Unit	Value Range
	PortXmitWait	infiniband_port_transmit_wait_total	The number of ticks during which the port had data to transmit but no data was sent during the entire tick (either because of insufficient credits or because of lack of arbitration).	Total count	Natural number
	PortXmitDiscards	infiniband_port_xmit_discards_total	Total number of outbound packets discarded by the port because the port is down or congested.	Total count	Natural number

Label Metrics

Table 10-4 Metric labels

Classification	Label	Description
Container metrics	modelarts_service	Service to which a container belongs, which can be notebook , train , or infer
	instance_name	Name of the pod to which the container belongs
	service_id	Instance or job ID displayed on the page, for example, cf55829e-9bd3-48fa-8071-7ae870dae93a for a development environment 9f322d5a-b1d2-4370-94df-5a87de27d36e for a training job
	node_ip	IP address of the node to which the container belongs
	container_id	Container ID
	cid	Cluster ID
	container_name	Container name
	project_id	Project ID of the account to which the user belongs

Classification	Label	Description
	user_id	User ID of the account to which the user who submits the job belongs
	pool_id	ID of a resource pool corresponding to a physical dedicated resource pool
	pool_name	Name of a resource pool corresponding to a physical dedicated resource pool
	logical_pool_id	ID of a logical subpool
	logical_pool_name	Name of a logical subpool
	gpu_uuid	UUID of the GPU used by the container
	gpu_index	Index of the GPU used by the container
	gpu_type	Type of the GPU used by the container
	account_name	Account name of the creator of a training, inference, or development environment task
	user_name	Username of the creator of a training, inference, or development environment task
	task_creation_time	Time when a training, inference, or development environment task is created
	task_name	Name of a training, inference, or development environment task
	task_spec_code	Specifications of a training, inference, or development environment task
	cluster_name	CCE cluster name
Node metrics	cid	ID of the CCE cluster to which the node belongs
	node_ip	IP address of the node
	host_name	Hostname of a node
	pool_id	ID of a resource pool corresponding to a physical dedicated resource pool
	project_id	Project ID of the user in a physical dedicated resource pool
	gpu_uuid	UUID of a node GPU
	gpu_index	Index of a node GPU
	gpu_type	Type of a node GPU

Classification	Label	Description
	device_name	Device name of an InfiniBand or RoCE network NIC
	port	Port number of the InfiniBand NIC
	physical_state	Status of each port on the InfiniBand NIC
	firmware_version	Firmware version of the IB NIC
	filesystem	NFS-mounted file system
	mount_point	NFS mount point
Diagnosis	cid	ID of the CCE cluster to which the node where the GPU resides belongs
	node_ip	IP address of the node where the GPU resides
	pool_id	ID of a resource pool corresponding to a physical dedicated resource pool
	project_id	Project ID of the user in a physical dedicated resource pool
	gpu_uuid	GPU UUID
	gpu_index	Index of a node GPU
	gpu_type	Type of a node GPU
	device_name	Name of a network device or disk device
	port	Port number of the InfiniBand NIC
	physical_state	Status of each port on the InfiniBand NIC
	firmware_version	Firmware version of the InfiniBand NIC

10.4 Using Grafana to View AOM Monitoring Metrics

10.4.1 Installing and Configuring Grafana

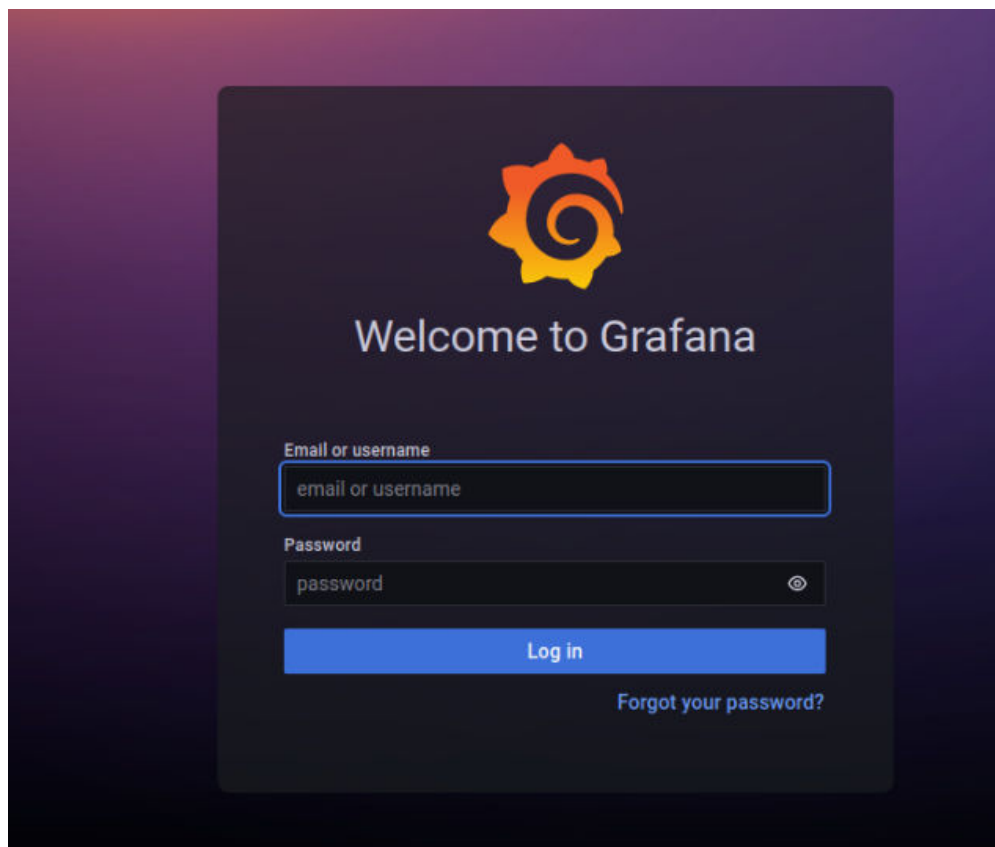
10.4.1.1 Installing and Configuring Grafana on Windows

Description

This section describes how to install and configure Grafana on a PC running Windows.

Procedure

1. Download the Grafana installation package.
Go to the [download link](#), click **Download the installer**, and wait until the download is successful.
2. Install Grafana.
Double-click the installation package and install Grafana as instructed.
3. In Windows Services Manager, enable Grafana.
4. Log in to Grafana.
Grafana runs on port 3000 by default. After you open <http://localhost:3000>, the Grafana login page is displayed. The default username and password for the first login are **admin**. After the login is successful, change the password as prompted.



10.4.1.2 Installing and Configuring Grafana on Linux

Description

This section describes how to install and configure Grafana on a PC running Linux.

Prerequisites

- An Ubuntu server that is accessible to the Internet is available. If no, the following conditions must be met:

- You have obtained an ECS. (You are advised to select 8 vCPUs or higher, Ubuntu image of 22.04 version, and 100 GB local storage.) For details, see [Purchasing a Custom ECS](#).
- You have purchased an EIP and bound it to the ECS. For details, see [Assigning an EIP and Binding It to an ECS](#).

Procedure

1. Log in to the ECS. Select a login method. For details, see .
2. Install libfontconfig1:

```
sudo apt-get install -y adduser libfontconfig1
```

The operation is successful if the following information is displayed.

```
root@ecs-9ec3:~# sudo apt-get install -y adduser libfontconfig1
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
adduser is already the newest version (3.118ubuntu5).
adduser set to manually installed.
libfontconfig1 is already the newest version (2.13.1-4.2ubuntu5).
libfontconfig1 set to manually installed.
The following packages were automatically installed and are no longer required:
  eatmydata libeatmydata1 libflashrom1 libftdi1-2 python-babel-localedata python3-babel python3-certifi python3-jinja2
  python3-json-pointer python3-jsonpatch python3-jsonschema python3-markupsafe python3-pyrsistent python3-requests python3-tz
  python3-urllib3
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 4 not upgraded.
```

3. Download the Grafana installation package:
wget https://dl.grafana.com/oss/release/grafana_9.3.6_amd64.deb --no-check-certificate

Download completed

```
root@ecs-9ec3:~# wget https://dl.grafana.com/oss/release/grafana_9.3.6_amd64.deb --no-check-certificate
--2023-03-07 10:22:12-- https://dl.grafana.com/oss/release/grafana_9.3.6_amd64.deb
Resolving dl.grafana.com (dl.grafana.com)... 151.101.42.217
Connecting to dl.grafana.com (dl.grafana.com)|151.101.42.217|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 89252050 (85M) [application/octet-stream]
Saving to: 'grafana_9.3.6_amd64.deb'

grafana_9.3.6_amd64.deb 100%[=====] 85.12M 379KB/s in 2m 21s
2023-03-07 10:24:36 (617 KB/s) - 'grafana_9.3.6_amd64.deb' saved [89252050/89252050]
```

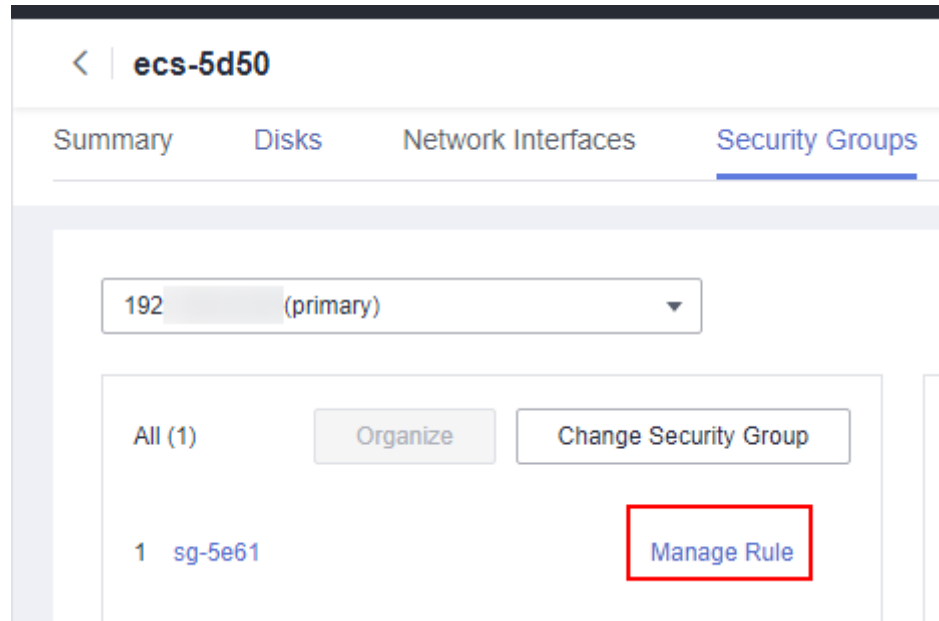
4. Install Grafana:
sudo dpkg -i grafana_9.3.6_amd64.deb

```
root@ecs-9ec3:~# sudo dpkg -i grafana_9.3.6_amd64.deb
Selecting previously unselected package grafana.
(Reading database ... 80788 files and directories currently installed.)
Preparing to unpack grafana_9.3.6_amd64.deb ...
Unpacking grafana (9.3.6) ...
Setting up grafana (9.3.6) ...
Adding system user `grafana' (UID 116) ...
Adding new user `grafana' (UID 116) with group `grafana' ...
Not creating home directory `/usr/share/grafana'.
### NOT starting on installation, please execute the following statements to configure grafana to start automatically using syst
emd
sudo /bin/systemctl daemon-reload
sudo /bin/systemctl enable grafana-server
### You can start grafana-server by executing
sudo /bin/systemctl start grafana-server
```

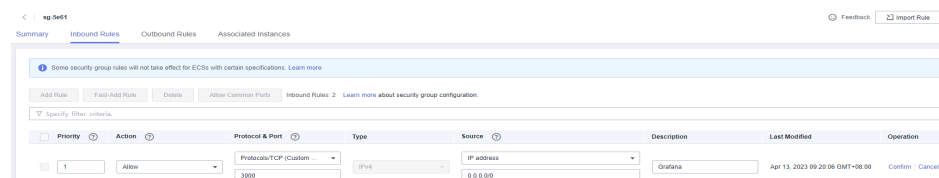
5. Start Grafana:
sudo /bin/systemctl start grafana-server
6. Access Grafana configurations on your local PC.

Ensure that an EIP has been bound to the ECS and the [security group](#) configuration is correct (the inbound traffic from TCP port 3000 and all outbound traffic are allowed). Configuration process:

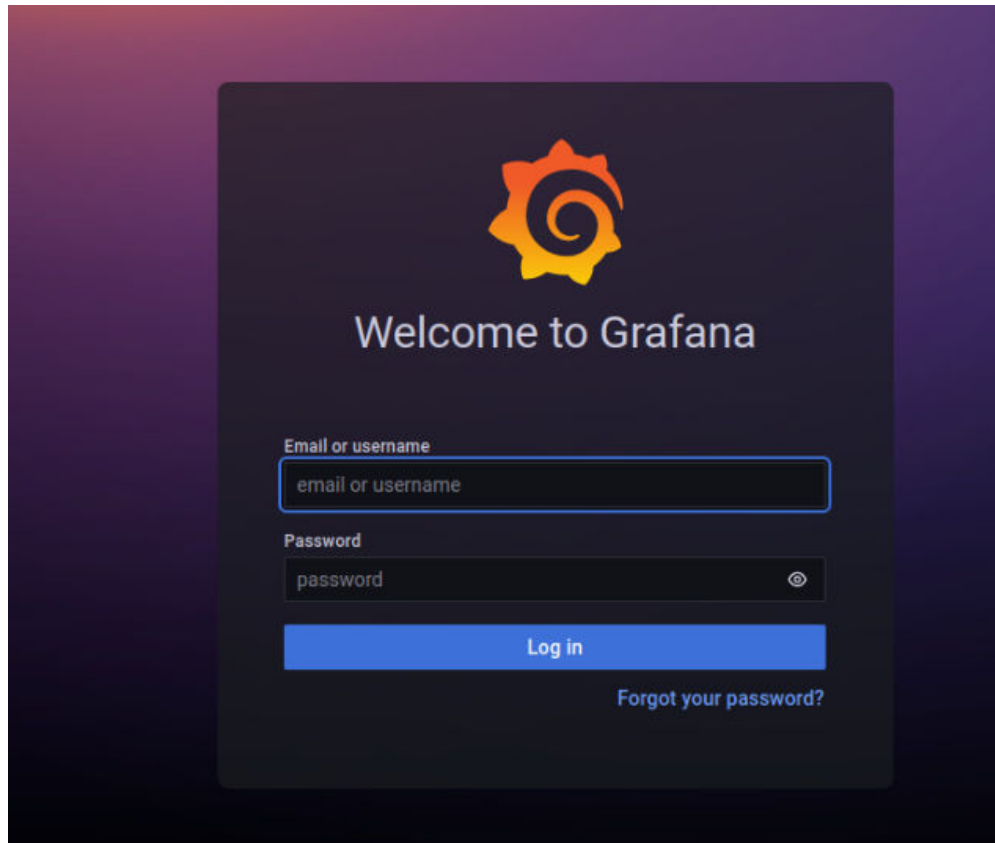
- a. Click the ECS name to go to the ECS details page. Then, click the **Security Groups** tab, and click **Manage Rule**.



- b. Click **Inbound Rules** and allow inbound traffic from TCP port 3000. By default, all outbound traffic is allowed.



7. Access **http://{EIP}:3000** in a browser. The default username and password for the first login are **admin**. After the login is successful, change the password as prompted.



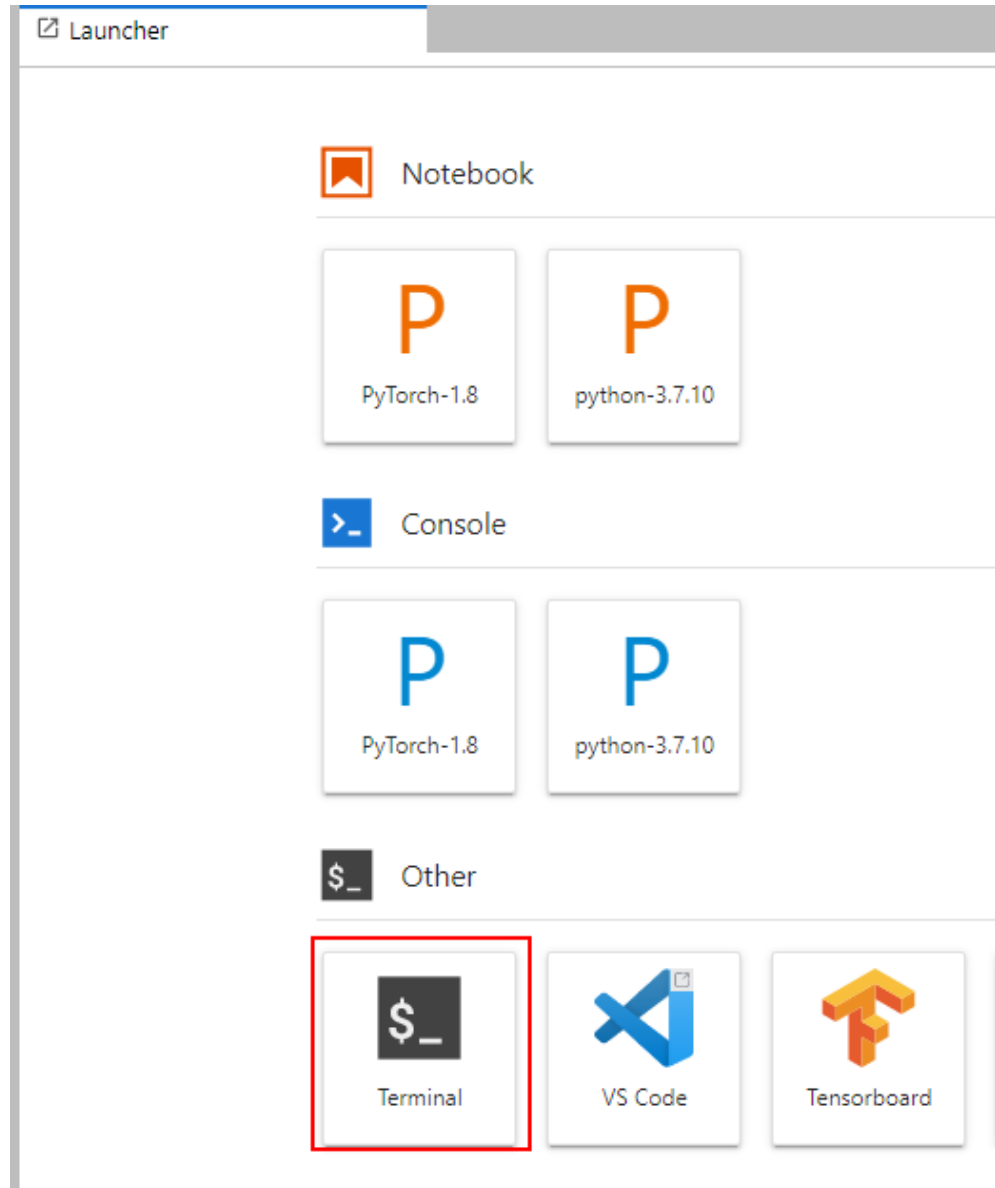
10.4.1.3 Installing and Configuring Grafana on a Notebook Instance

Description

This section describes how to install and configure Grafana in ModelArts Standard notebook instances.

Prerequisites

- A running CPU- or GPU-based notebook instance is available.
- A terminal is opened.



Procedure

1. Run the following commands in sequence in your terminal to download and install Grafana:

```
mkdir -p /home/ma-user/work/grf
cd /home/ma-user/work/grf
wget https://dl.grafana.com/oss/release/grafana-9.1.6.linux-amd64.tar.gz
tar -zxvf grafana-9.1.6.linux-amd64.tar.gz
```

```
(PyTorch-1.8) [ma-user work]$ mkdir -p /home/ma-user/work/grf
(PyTorch-1.8) [ma-user work]$ cd /home/ma-user/work/grf
(PyTorch-1.8) [ma-user grf]$ wget https://dl.grafana.com/oss/release/grafana-9.1.6.linux-amd64.tar.gz
--2023-03-08 15:53:41-- https://dl.grafana.com/oss/release/grafana-9.1.6.linux-amd64.tar.gz
Resolving proxy.modelarts.com (proxy.modelarts.com)... 192.168.6.3
Connecting to proxy.modelarts.com (proxy.modelarts.com)|192.168.6.3|:80... connected.
Proxy request sent, awaiting response... 200 OK
Length: 81857482 (77M) [application/x-tar]
Saving to: 'grafana-9.1.6.linux-amd64.tar.gz.1'
grafana-9.1.6.linux-amd64.tar.gz.1 5K[====>] 4.41M 57.0KB/s eta 8m 19s
```

2. Register Grafana with jupyter-server-proxy.
 - a. Run the following commands in your terminal:

```
mkdir -p /home/ma-user/.local/etc/jupyter
vi /home/ma-user/.local/etc/jupyter/jupyter_notebook_config.py
```

```
(PyTorch-1.8) [ma-user grf]$mkdir -p /home/ma-user/.local/etc/jupyter
(PyTorch-1.8) [ma-user grf]$vi /home/ma-user/.local/etc/jupyter/jupyter_notebook_config.py
```

- b. In `jupyter_notebook_config.py`, add the following code, press **Esc** to exit, and type `:wq` to save the changes:

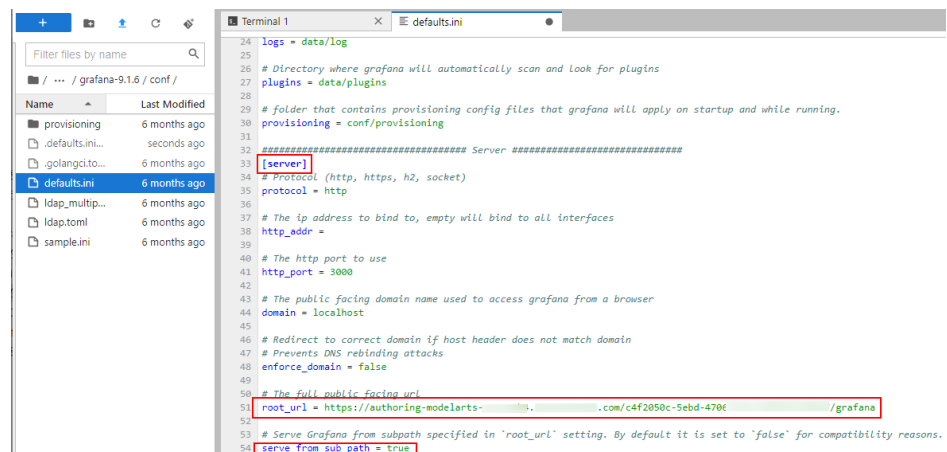
```
c.ServerProxy.servers = {
    'grafana': {
        'command': ['/home/ma-user/work/grf/grafana-9.1.6/bin/grafana-server', '--
homeopath', '/home/ma-user/work/grf/grafana-9.1.6', 'web'],
        'timeout': 1800,
        'port': 3000
    }
}
```

NOTE

If `jupyter_notebook_config.py` (path: `/home/ma-user/.local/etc/jupyter/jupyter_notebook_config.py`) contains the `c.ServerProxy.servers` field, add the corresponding key-value pair.

3. Modify the URL for accessing Grafana in JupyterLab.
- a. In the navigation pane on the left, open the `vi /home/ma-user/work/grf/grafana-9.1.6/conf/defaults.ini` file.
- b. Change the `root_url` and `serve_from_sub_path` fields in `[server]`.

Figure 10-2 Modifying the `defaults.ini` file



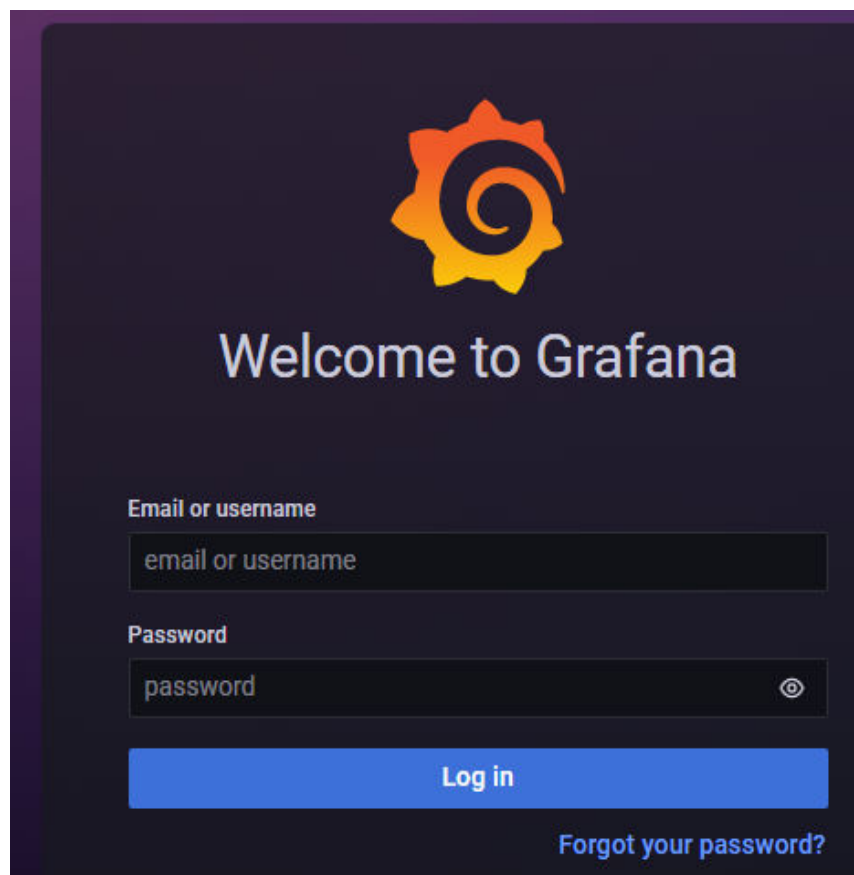
In the file:

- The value of `root_url` is in the format of `https://{JupyterLab domain name}/{Instance ID}/grafana`. You can obtain the domain name and instance ID from the address box of the JupyterLab page.

- Set `Serve_from_sub_path` to `true`.

4. Save the image of the notebook instance.
- a. Log in to the ModelArts console. In the navigation pane on the left, choose **Development Workspace > Notebook**. On the displayed page, locate the target instance, and choose **More > Save Image** in the **Operation** column.

- b. In the displayed dialog box, configure the organization, image name, image version, and description, and click **OK**.
 - c. The image will be saved as a snapshot, and it will take about 5 minutes. During this period of time, do not perform any operations on the instance.
 - d. After the image is saved, the instance status changes to **Running**. Then, restart the notebook instance.
5. Open the Grafana page.
Open a browser window and type the value of **root_url** configured in **3** in the address box. If the Grafana login page is displayed, Grafana is installed and configured in the notebook instance. The default username and password for the first login are **admin**. After the login is successful, change the password as prompted.



10.4.2 Configuring a Grafana Data Source

Before viewing ModelArts monitoring data on Grafana, configure the data source.

Prerequisites

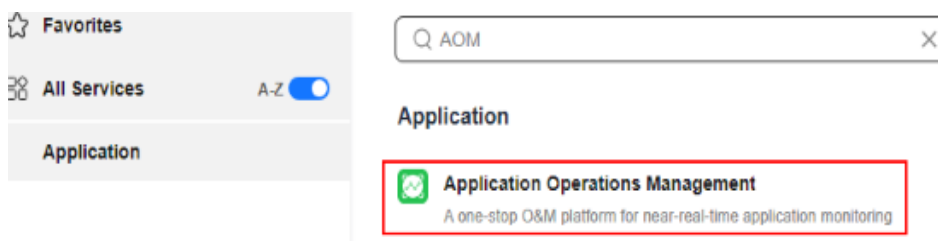
Grafana has been installed.

Procedure

1. Obtain the Grafana data source configuration code.

- a. Log in to the AOM console.

Figure 10-3 AOM console



- b. In the navigation pane on the left, choose **Prometheus Monitoring** > **Instances**. Then, click the **Prometheus_AOM_Default** instance.

Figure 10-4 Prometheus_AOM_Default

Prometheus Instance	Instance Type	Enterprise Project
Prometheus_AOM_Default	default	default

- c. In the **Settings** tab, obtain the Grafana data source configuration code of the instance in the **Grafana Data Source Info** area.

Grafana Data Source Info

Intranet Public Network

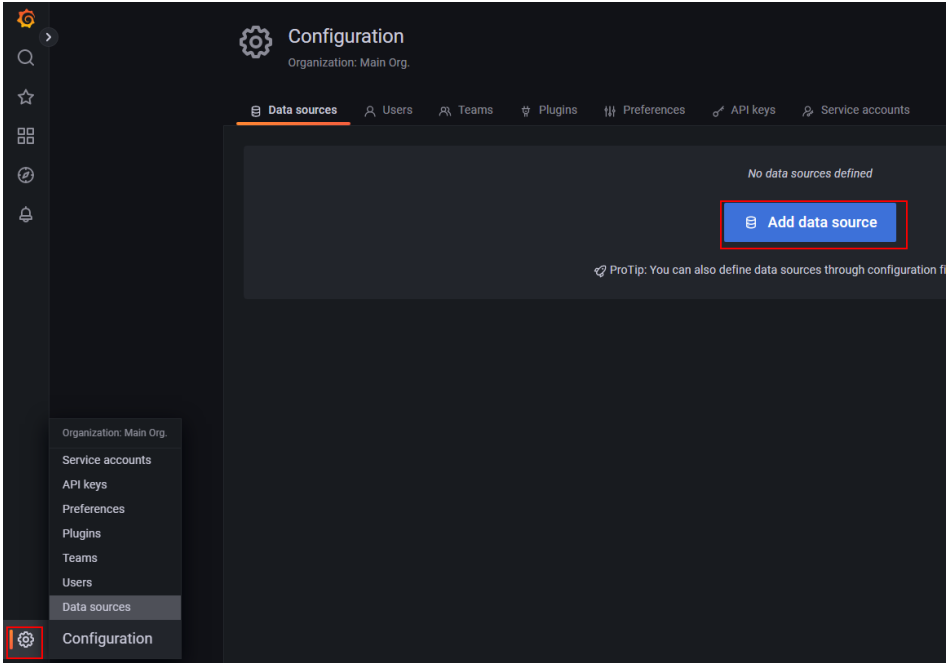
HTTP URL

Username

Password

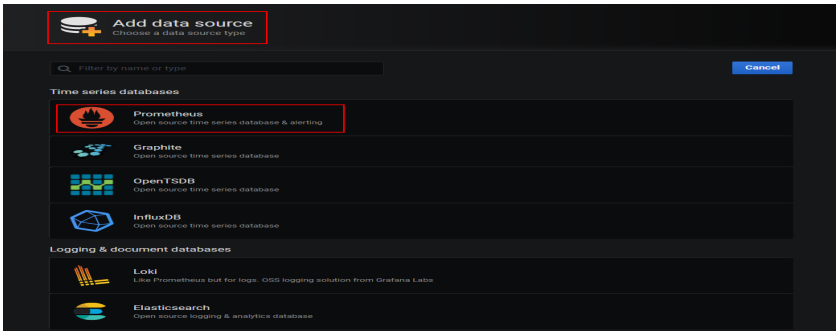
2. Add a data source to Grafana.
 - a. Log in to Grafana. The default username and password for the first login are **admin**. After the login is successful, change the password as prompted.
 - b. In the navigation pane, choose **Configuration** > **Data Sources**. Then, click **Add data source**.

Figure 10-5 Configuring Grafana



- c. Click **Prometheus** to access the configuration page.

Figure 10-6 Prometheus configuration page



- d. Configure parameters as shown in the following figure.

Figure 10-7 Configuring a Grafana data source

Name ⓘ PrometheusDefault ☐

HTTP

URL ⓘhttps://

AccessServer (default) ▾Help >

Allowed cookies ⓘNew tag (enter key to add)

Timeout ⓘTimeout in seconds

Auth

Basic auth ☒With Credentials ⓘ ☐

TLS Client Auth ☐With CA Cert ⓘ ☐

Skip TLS Verify ☒

Forward OAuth Identity ⓘ ☐

Basic Auth Details

Useruser

PasswordconfiguredReset

 NOTE

The actual Grafana version varies depending on the installation method. [Figure 10-7](#) is only an example.

Table 10-5 Parameters

Parameter	Description
Name	Enter a name.
URL	Set this parameter to the HTTP URL obtained in step 1.
Basic auth	Enable it.
Skip TLS Verify	Enable it.
User	Set this parameter to the username obtained in step a.
Password	Set this parameter to the password obtained in step a.

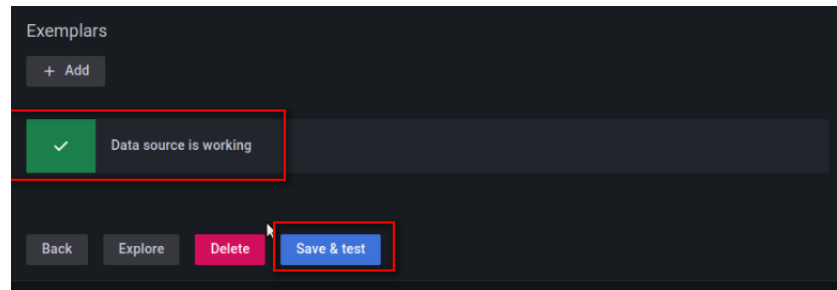
Issue 01 (2025-08-29)

Copyright © Huawei Cloud Computing Technologies Co., Ltd.

1058

- e. After the configuration, click **Save & test**. If the message **Data source is working** is displayed, the data source is configured.

Figure 10-8 Data source added



10.4.3 Configuring a Dashboard to View Metric Data

In Grafana, you can customize dashboards for various views. ModelArts also provides configuration templates for clusters. This section describes how to configure a dashboard by using a ModelArts template or creating a dashboard. For more usage, see [Grafana tutorials](#).

Preparations

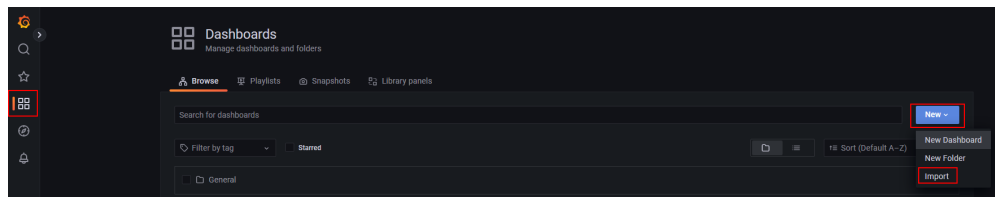
ModelArts provides templates for cluster view, node view, user view, task view, and task details view. These templates can be downloaded from Grafana official documents. You can import and use them on Dashboards.

Table 10-6 Template download URLs

Template	Download URL
Cluster view	https://cnnorth4-modelarts-sdk.obs.cn-north-4.myhuaweicloud.com/metrics/grafana/dashboards/ModelArts-Cluster-View.json
Node view	https://cnnorth4-modelarts-sdk.obs.cn-north-4.myhuaweicloud.com/metrics/grafana/dashboards/ModelArts-Node-View.json
User view	https://cnnorth4-modelarts-sdk.obs.cn-north-4.myhuaweicloud.com/metrics/grafana/dashboards/ModelArts-User-View.json
Task view	https://cnnorth4-modelarts-sdk.obs.cn-north-4.myhuaweicloud.com/metrics/grafana/dashboards/ModelArts-Task-View.json
Task details view	https://cnnorth4-modelarts-sdk.obs.cn-north-4.myhuaweicloud.com/metrics/grafana/dashboards/ModelArts-Task-Detail-View.json

Using a ModelArts Template to View Metrics

1. Open **Dashboards** and choose **New > Import**.



2. Import the dashboard template.

Copy the template download URL provided in [Preparations](#) to a web browser and copy the content of the JSON file. Paste the content to the dashboard template and click **Load**.

Figure 10-9 Copying the content of the JSON file

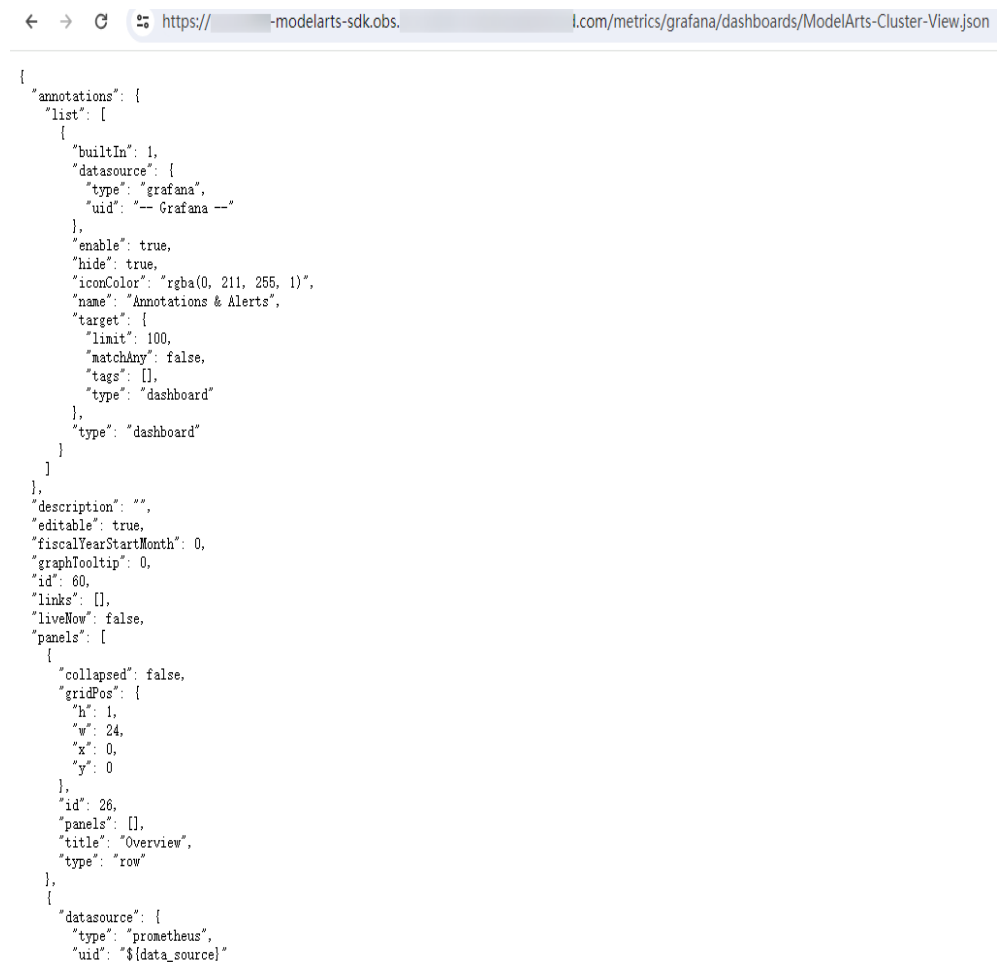
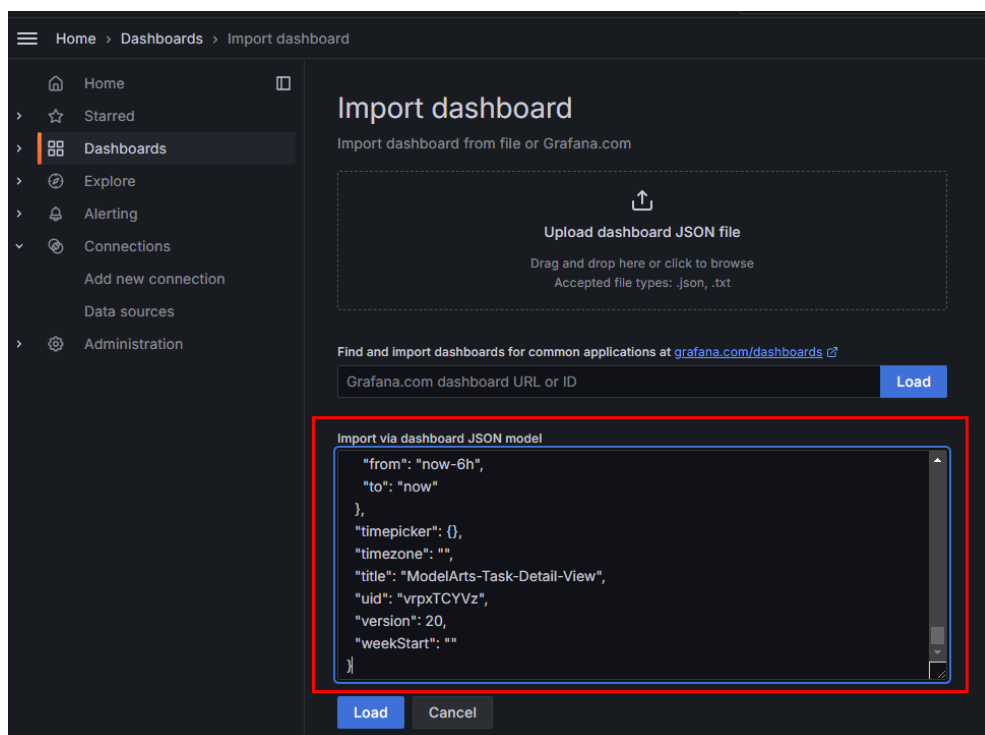
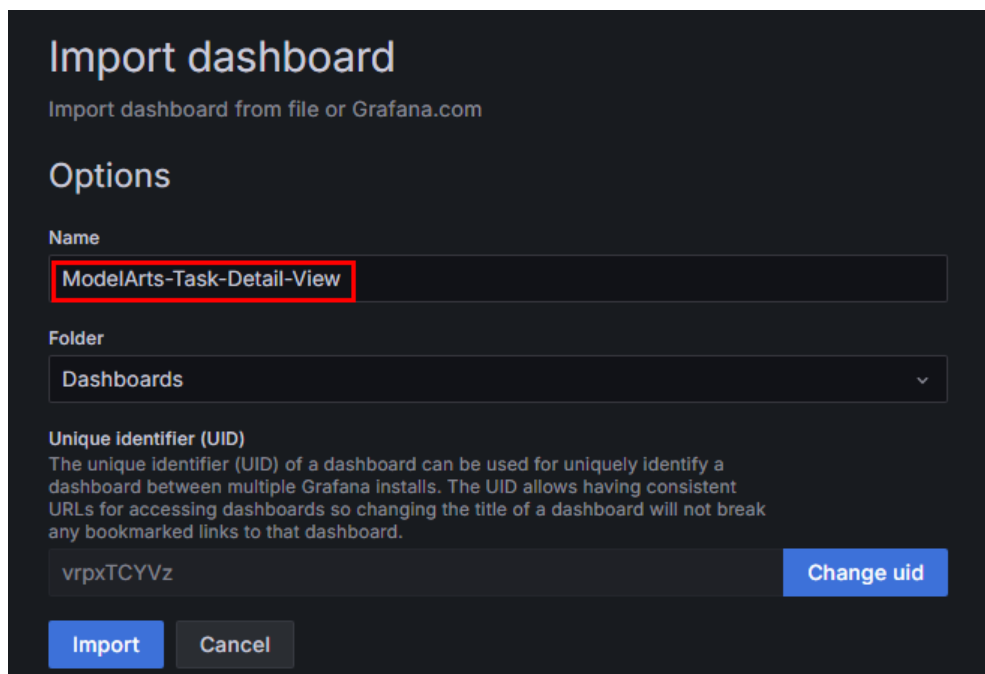


Figure 10-10 Pasting the content of the JSON file to the dashboard template



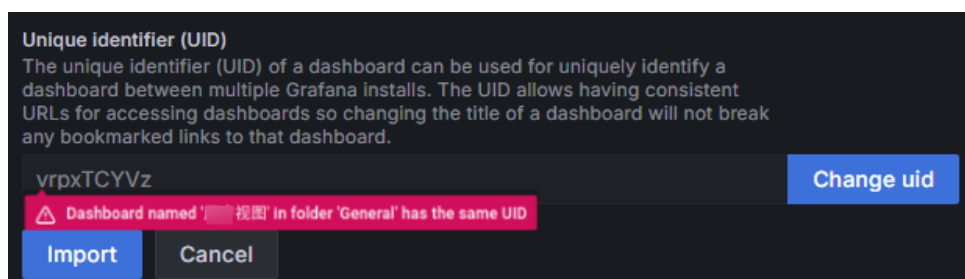
3. Change the view name and click **Import**.

Figure 10-11 Changing the view name

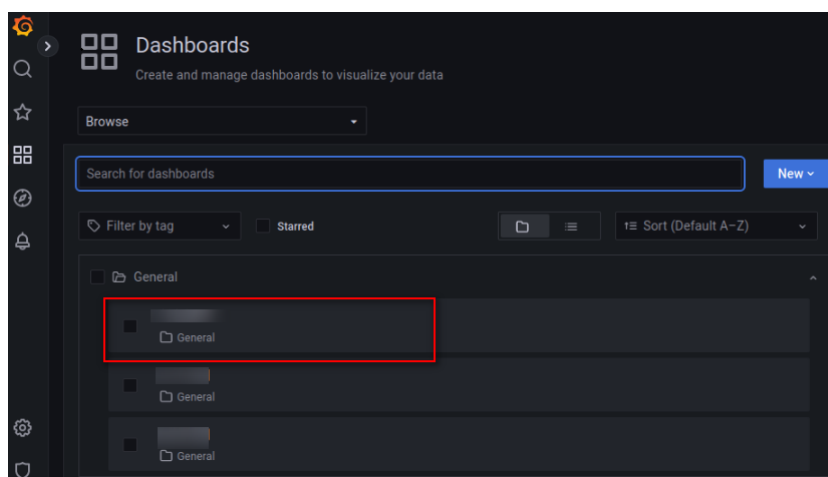


Note: If a message is displayed, indicating that the UID is duplicate, click **Change uid**, change the UID in the JSON file, and click **Import**.

Figure 10-12 Changing the UID



4. After the import, view the imported views in **Dashboards**. Then, click a view to open the monitoring page.

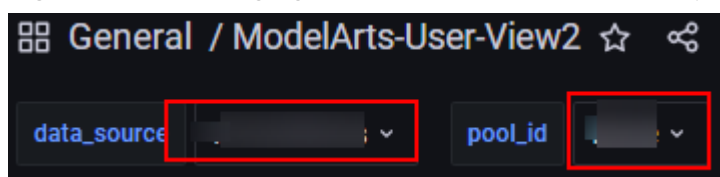


5. Use the template.

After the import is successful, you can click the template to view its details. This section introduces some commonly-used functions.

- Changing the data source and resource pool

Figure 10-13 Changing the data source and resource pool



Click the area marked by the red box. A drop-down list will appear. From there, you can change the data source and the resource pool.

- Refreshing data



Click the refresh button in the upper right corner to refresh all data on the dashboard. The data on each panel is also updated.

- Changing the automatic refresh time

The default refresh interval of a template is 15 minutes. If you need to update the interval, change the value from the drop-down list box in the upper right corner.

- Changing the time range for obtaining dashboard data

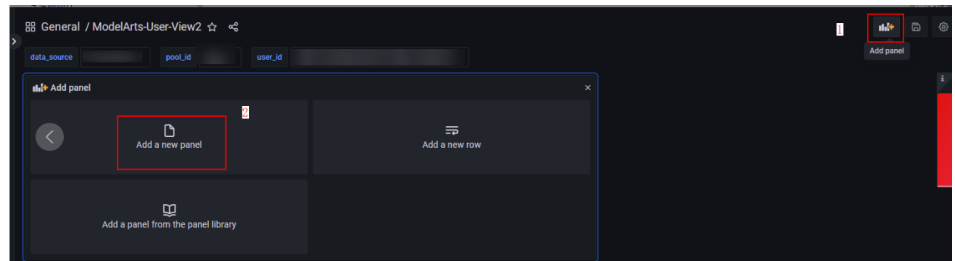
Figure 10-14 Changing the time range for obtaining data



Click the button in the upper right corner to change time range for obtaining data. This time range affects all panels except those with a fixed time.

- Adding a panel

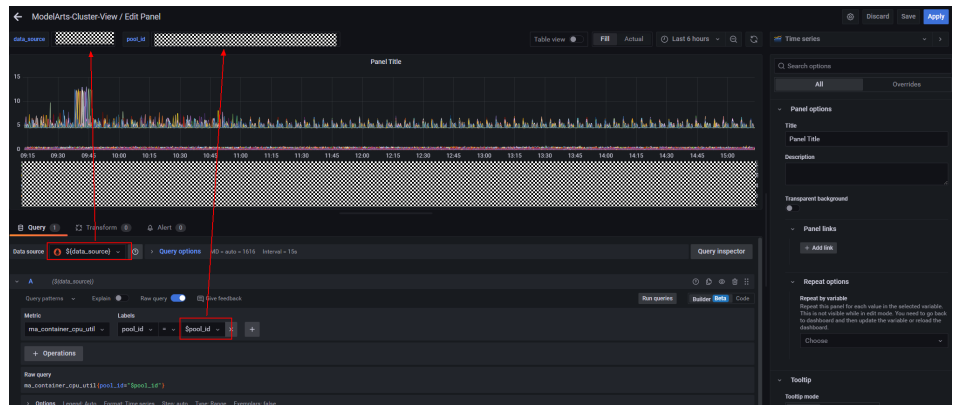
Figure 10-15 Adding a panel



Click the + icon in the upper right corner to add a panel.

After a panel is added, you can obtain the data in the panel. Configure the data source and resource pool as follows to use the current dashboard settings.

Figure 10-16 Using the current dashboard settings



Creating a Dashboard to View Metrics

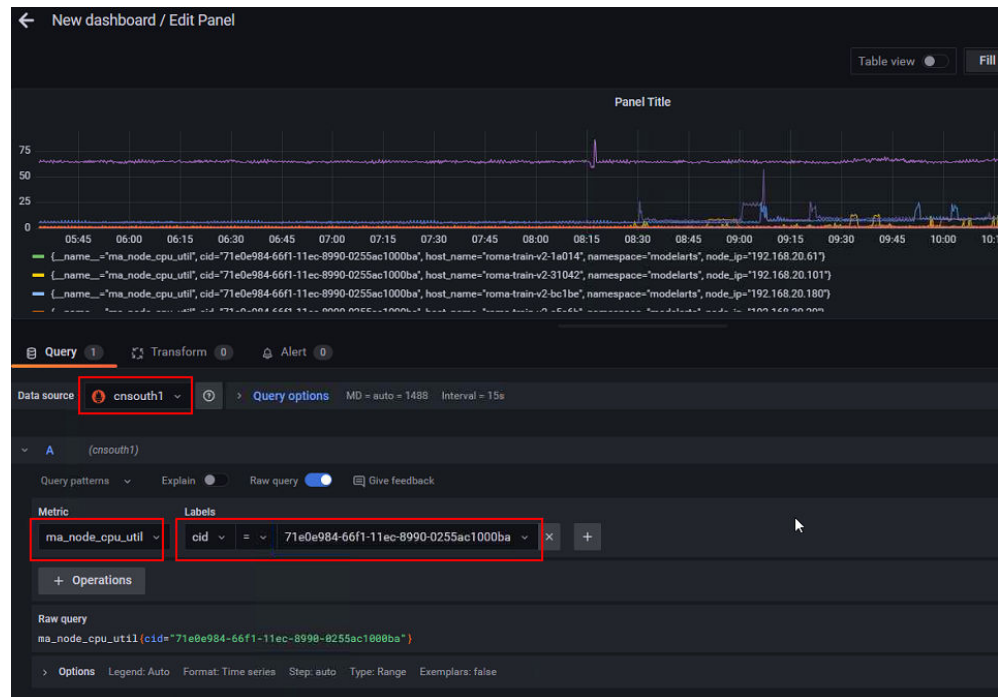
1. Open **Dashboards**, click **New**, and choose **New Dashboard**.
2. Click **Add a new panel**.
3. On the **New dashboard / Edit Panel** page, set the following parameters:

Data source: [Configured Grafana data source](#)

Metric: Metric name. You can obtain the metric to be queried by referring to [Table 10-1](#), [Table 10-2](#), and [Table 10-3](#).

Labels: Used for filtering the metric. For details, see and [Table 10-4](#).

Figure 10-17 Creating a dashboard to view metrics



11

Using CTS to Audit ModelArts Standard

11.1 ModelArts Standard Key Operations Traced by CTS

ModelArts can connect to CTS. With CTS, you can obtain operations associated with ModelArts Standard for later query, audit, and backtrack operations.

Prerequisites

CTS has been enabled. For details, see [Cloud Trace Service User Guide](#).

Key Data Management Operations Traced by CTS

Table 11-1 Key data management operations traced by CTS

Operation	Resource Type	Trace
Creating a dataset	Dataset	createDataset
Deleting a dataset	Dataset	deleteDataset
Updating a dataset	Dataset	updateDataset
Publishing a dataset version	Dataset	publishDatasetVersion
Deleting a dataset version	Dataset	deleteDatasetVersion
Synchronizing the data source	Dataset	syncDataSource
Exporting a dataset	Dataset	exportDataFromDataset
Creating an auto labeling task	Dataset	createAutoLabelingTask
Creating an auto grouping task	Dataset	createAutoGroupingTask

Operation	Resource Type	Trace
Creating an auto deployment task	Dataset	createAutoDeployTask
Importing samples to a dataset	Dataset	importSamplesToDataset
Creating a dataset label	Dataset	createLabel
Updating a dataset label	Dataset	updateLabel
Deleting a dataset label	Dataset	deleteLabel
Deleting a dataset label and its samples	Dataset	deleteLabelWithSamples
Adding samples	Dataset	uploadSamples
Deleting samples	Dataset	deleteSamples
Stopping an auto labeling task	Dataset	stopTask
Creating a team labeling task	Dataset	createWorkforceTask
Deleting a team labeling task	Dataset	deleteWorkforceTask
Starting the acceptance of a team labeling task	Dataset	startWorkforceSampling-Task
Approving, rejecting, or canceling the acceptance of a team labeling task	Dataset	updateWorkforceSamplingTask
Submitting sample review comments for an acceptance task	Dataset	acceptSamples
Adding a label to a sample	Dataset	updateSamples
Sending an email to team labeling members	Dataset	sendEmails
Starting a team labeling task as the team manager	Dataset	startWorkforceTask
Updating a team labeling task	Dataset	updateWorkforceTask
Adding a label to a team-labeled sample	Dataset	updateWorkforceTask-Samples

Operation	Resource Type	Trace
Reviewing team labeling results	Dataset	reviewSamples
Creating a labeling team member	Workforce	createWorker
Updating labeling team members	Workforce	updateWorker
Deleting a labeling team member	Workforce	deleteWorker
Deleting labeling team members in batches	Workforce	batchDeleteWorker
Creating a labeling team	Workforce	createWorkforce
Updating a labeling team	Workforce	updateWorkforce
Deleting a labeling team	Workforce	deleteWorkforce
Automatically creating an IAM agency	IAM	createAgency
Logging in to the labeling console as a team labeling member	labelConsoleWorker	workerLoginLabelConsole
Logging out of the labeling console as a team labeling member	labelConsoleWorker	workerLogOutLabelConsole
Changing the password for logging in to the labeling console as a team labeling member	labelConsoleWorker	workerChangePassword
Handling the issue that the password for logging in to the labeling console as a team labeling member is lost	labelConsoleWorker	workerForgetPassword
Resetting the password for logging in to the labeling console through the URL as a team labeling member	labelConsoleWorker	workerResetPassword

Key Development Environment Operations Traced by CTS

Table 11-2 Key development environment operations traced by CTS

Operation	Resource Type	Trace
Creating a notebook instance	Notebook	createNotebook
Deleting a notebook instance	Notebook	deleteNotebook
Opening a notebook instance	Notebook	openNotebook
Starting a notebook instance	Notebook	startNotebook
Stopping a notebook instance	Notebook	stopNotebook
Updating a notebook instance	Notebook	updateNotebook
Querying notebook instances	Notebook	listNotebook
Querying all notebook instances	Notebook	listallNotebook
Obtaining details about a notebook instance	Notebook	getNotebook
Querying notebook tags in a project	NotebookTmsTags	listNotebookTmsTags
Querying supported images	NotebookImage	listNotebookImage
Querying user image information	NotebookImageGroups	listNotebookImageGroups
Querying image details	NotebookImage	getNotebookImage

Key Training Job Operations Traced by CTS

Table 11-3 Key training job operations traced by CTS

Operation	Resource Type	Trace
Creating a training job	ModelArtsTrainJob	createModelArtsTrainJob
Creating a training job version	ModelArtsTrainJob	createModelArtsTrainVersion

Operation	Resource Type	Trace
Stopping a training job	ModelArtsTrainJob	stopModelArtsTrainVersion
Modifying the description of a training job	ModelArtsTrainJob	updateModelArtsTrainDesc
Deleting a training job version	ModelArtsTrainJob	deleteModelArtsTrainVersion
Deleting a training job	ModelArtsTrainJob	deleteModelArtsTrainJob
Configuring a training job	ModelArtsTrainConfig	createModelArtsTrainConfig
Modifying training job configurations	ModelArtsTrainConfig	updateModelArtsTrainConfig
Deleting training job configurations	ModelArtsTrainConfig	deleteModelArtsTrainConfig
Creating a visualization job	ModelArtsTensorboardJob	createModelArtsTensorboardJob
Deleting a visualization job	ModelArtsTensorboardJob	deleteModelArtsTensorboardJob
Modifying the description of a visualization job	ModelArtsTensorboardJob	updateModelArtsTensorboardDesc
Stopping a visualization job	ModelArtsTensorboardJob	stopModelArtsTensorboardJob
Restarting a visualization job	ModelArtsTensorboardJob	restartModelArtsTensorboardJob

Key Model Management Operations Traced by CTS

Table 11-4 Key model management operations traced by CTS

Operation	Resource Type	Trace
Creating a model	Model	addModel
Updating a model	Model	updateModel
Deleting a model	Model	deleteModel
Creating a model conversion task	Convert	addConvert

Operation	Resource Type	Trace
Updating a model conversion task	Convert	updateConvert
Deleting a model conversion task	Convert	deleteConvert

Key Service Management Operations Traced by CTS

Table 11-5 Key service management operations traced by CTS

Operation	Resource Type	Trace
Deploying a service	Service	addService
Deleting a service	Service	deleteService
Updating a service	Service	updateService
Starting or stopping a service	Service	startOrStopService
Adding an access key	Service	addAkSk
Deleting an access key	Service	deleteAkSk
Creating a dedicated resource pool	Cluster	createCluster
Deleting a dedicated resource pool	Cluster	deleteCluster
Adding a node to a dedicated resource pool	Cluster	addClusterNode
Deleting a node from a dedicated resource pool	Cluster	deleteClusterNode
Obtaining the result of creating a dedicated resource pool	Cluster	createClusterResult

Key AI Gallery Operations Traced by CTS

Table 11-6 Key AI Gallery operations traced by CTS

Operation	Resource Type	Trace
Publishing an asset	ModelArts_Market	create_content

Operation	Resource Type	Trace
Modifying asset information	ModelArts_Market	modify_content
Publishing an asset version	ModelArts_Market	add_version
Subscribing to an asset	ModelArts_Market	subscription_content
Removing an asset from favorites	ModelArts_Market	cancel_star_content
Liking an asset	ModelArts_Market	like_content
Unliking an asset	ModelArts_Market	cancel_like_content
Publishing an activity	ModelArts_Market	publish_activity
Signing up an activity	ModelArts_Market	regist_activity
Modifying user information	ModelArts_Market	update_user

Key Resource Management Operations Traced by CTS

Table 11-7 Key resource management operations traced by CTS

Operation	Resource Type	Trace
Creating a resource pool	PoolV2	CreatePoolV2
Deleting a resource pool	PoolV2	DeletePoolV2
Updating a resource pool	PoolV2	UpdatePoolV2
Creating a network	NetworksV1	CreateNetworksV1
Deleting a network	NetworksV1	DeleteNetworksV1
Updating a network	NetworksV1	UpdateNetworksV1

Console Backend Key Operations Traced by CTS

Table 11-8 Console Backend key operations traced by CTS

Operation	Resource Type	Trace
Creating an agency	ModelArtsConsoleBack-end	createAgency

Operation	Resource Type	Trace
Updating the permissions of an agency	ModelArtsConsoleBack-end	updateAgencyGrants
Deleting an agency	ModelArtsConsoleBack-end	deleteAgency
Querying an agency	ModelArtsConsoleBack-end	queryAgency
Querying agency permissions	ModelArtsConsoleBack-end	listAgencyGrants
Obtaining agency information	ModelArtsConsoleBack-end	getAgencyInfo
Querying agency quotas	ModelArtsConsoleBack-end	getAgencyQuotas
Adding an access authorization	ModelArtsConsoleBack-end	addAuthorization
Deleting an access authorization	ModelArtsConsoleBack-end	deleteAuthorization
Querying the authorization list	ModelArtsConsoleBack-end	listAuthorization
Querying a feature switch	ModelArtsConsoleBack-end	showFeature
Adding a feature switch	ModelArtsConsoleBack-end	showFeature
Creating an OBS directory	ModelArtsConsoleBack-end	createOBSFolder
Obtaining the bucket list	ModelArtsConsoleBack-end	listBuckets
Obtaining the bucket object list	ModelArtsConsoleBack-end	listBucketObjects
Obtaining bucket encryption information	ModelArtsConsoleBack-end	getBucketEncryption
Buying a package	ModelArtsConsoleBack-end	buyPackages
Subscribing to an AI cloud package	ModelArtsConsoleBack-end	buyAiPackage
Clearing resources	ModelArtsConsoleBack-end	cleanResource

You can view audit logs of the operations of buying packages and subscribing to AI cloud packages in the CN North-Beijing4, CN North-Beijing1, and CN East-Shanghai1 regions.

11.2 Viewing ModelArts Standard Audit Logs

After CTS is enabled, CTS starts recording operations on ModelArts. The CTS console stores the operation records generated in the last seven days. This section describes how to view operation records of the last seven days on the CTS console.

Procedure

1. Log in to the CTS console.
2. Click  in the upper left corner and select a region.
3. In the navigation pane on the left, choose **Trace List**.
4. Specify filters as needed. You can query traces using a combination of the following filters:
 - **Trace Source, Resource Type, and Search By:**
Select a filter from the drop-down list.
When you select **Trace Name**, you need to enter a specific trace name.
When you select **Resource ID**, you need to enter a specific resource ID.
When you select **Resource Name**, you need to enter a specific resource name.
 - **Operator:** Select a specific operator (a user other than tenant).
 - **Trace Status:** Select **All trace statuses**, **Normal**, **Warning**, or **Incident**.
 - **Operation Time:** You can query traces generated during any time range in the last seven days.
5. Click  on the left of a trace to expand its details.
6. Locate the target trace and click **View Trace** in the **Operation** column. In the displayed **View Trace** dialog box, view the trace structure details.
For details about the CTS trace structure, see [CTS User Guide](#).