

ModelArts

DevEnviron

Issue 01
Date 2023-05-16



Copyright © Huawei Cloud Computing Technologies Co., Ltd. 2024. 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 Introduction to DevEnviron.....	1
2 Application Scenarios.....	5
3 Managing Notebook Instances.....	6
3.1 Creating a Notebook Instance.....	6
3.2 Accessing a Notebook Instance.....	13
3.3 Searching for, Starting, Stopping, or Deleting a Notebook Instance.....	14
3.4 Changing a Notebook Instance Image.....	16
3.5 Changing the Flavor of a Notebook Instance.....	16
3.6 Selecting Storage in DevEnviron.....	17
3.7 Dynamically Mounting an OBS Parallel File System.....	21
3.8 Dynamically Expanding EVS Disk Capacity.....	23
3.9 Modifying the SSH Configuration for a Notebook Instance.....	24
3.10 Viewing the Notebook Instances of All IAM Users Under One Tenant Account.....	26
3.11 Viewing Notebook Events.....	28
3.12 Notebook Cache Directory Alarm Reporting.....	32
4 JupyterLab.....	38
4.1 Operation Process in JupyterLab.....	38
4.2 JupyterLab Overview and Common Operations.....	39
4.3 Code Parametrization Plug-in.....	47
4.4 Using ModelArts SDK.....	49
4.5 Using the Git Plug-in.....	50
4.6 Visualized Model Training.....	55
4.6.1 Introduction to Training Job Visualization.....	55
4.6.2 MindInsight Visualization Jobs.....	56
4.6.3 TensorBoard Visualization Jobs.....	62
4.7 Uploading and Downloading Data in Notebook.....	69
4.7.1 Uploading Files to JupyterLab.....	69
4.7.1.1 Scenarios.....	69
4.7.1.2 Uploading Files from a Local Path to JupyterLab.....	69
4.7.1.2.1 Upload Scenarios and Entries.....	69
4.7.1.2.2 Uploading a Local File Less Than 100 MB to JupyterLab.....	71
4.7.1.2.3 Uploading a Local File with a Size Ranging from 100 MB to 5 GB to JupyterLab.....	72

4.7.1.2.4 Uploading a Local File Larger Than 5 GB to JupyterLab.....	75
4.7.1.3 Cloning an Open-Source Repository in GitHub.....	77
4.7.1.4 Uploading OBS Files to JupyterLab.....	78
4.7.1.5 Uploading Remote Files to JupyterLab.....	81
4.7.2 Downloading a File from JupyterLab to a Local Path.....	82
5 Local IDE.....	85
5.1 Operation Process in a Local IDE.....	85
5.2 Local IDE (PyCharm).....	86
5.2.1 Connecting to a Notebook Instance Through PyCharm Toolkit.....	86
5.2.1.1 PyCharm Toolkit.....	86
5.2.1.2 Downloading and Installing PyCharm Toolkit.....	87
5.2.1.3 Connecting to a Notebook Instance Through PyCharm Toolkit.....	88
5.2.2 Manually Connecting to a Notebook Instance Through PyCharm.....	95
5.2.3 Submitting a Training Job Using PyCharm Toolkit.....	101
5.2.3.1 Submitting a Training Job (New Version).....	101
5.2.3.2 Stopping a Training Job.....	105
5.2.3.3 Viewing Training Logs.....	106
5.2.4 Uploading Data to a Notebook Instance Using PyCharm.....	106
5.3 Local IDE (VS Code).....	108
5.3.1 Connecting to a Notebook Instance Through VS Code.....	108
5.3.2 Installing VS Code.....	108
5.3.3 Connecting to a Notebook Instance Through VS Code Toolkit.....	109
5.3.4 Manually Connecting to a Notebook Instance Through VS Code.....	115
5.3.5 Remotely Debugging in VS Code.....	121
5.3.6 Uploading and Downloading Files in VS Code.....	123
5.4 Local IDE (Accessed Using SSH).....	125
6 ModelArts CLI Command Reference.....	132
6.1 ModelArts CLI Overview.....	132
6.2 (Optional) Installing ma-cli Locally.....	134
6.3 Autocompletion for ma-cli Commands.....	135
6.4 ma-cli Authentication.....	136
6.5 ma-cli Image Building Command.....	138
6.5.1 ma-cli Image Building Command.....	138
6.5.2 Obtaining an Image Creation Template.....	139
6.5.3 Loading an Image Creation Template.....	140
6.5.4 Obtaining Registered ModelArts Images.....	141
6.5.5 Creating an Image in ModelArts Notebook.....	143
6.5.6 Obtaining Image Creation Caches in ModelArts Notebook.....	145
6.5.7 Clearing Image Creation Caches in ModelArts Notebook.....	146
6.5.8 Registering SWR Images with ModelArts Image Management.....	147
6.5.9 Deregistering a Registered Image from ModelArts Image Management.....	149
6.5.10 Debugging an SWR Image on an ECS.....	149

6.6 Using the ma-cli ma-job Command to Submit a ModelArts Training Job.....	150
6.6.1 ma-cli ma-job Command Overview.....	150
6.6.2 Obtaining ModelArts Training Jobs.....	151
6.6.3 Submitting a ModelArts Training Job.....	153
6.6.4 Obtaining ModelArts Training Job Logs.....	158
6.6.5 Obtaining ModelArts Training Job Events.....	159
6.6.6 Obtaining ModelArts AI Engines for Training.....	160
6.6.7 Obtaining ModelArts Resource Specifications for Training.....	161
6.6.8 Stopping a ModelArts Training Job.....	162
6.7 Using the ma-cli dli-job Command to Submit a DLI Spark Job.....	163
6.7.1 Overview.....	163
6.7.2 Querying DLI Spark Jobs.....	164
6.7.3 Submitting a DLI Spark Job.....	166
6.7.4 Querying DLI Spark Run Logs.....	171
6.7.5 Querying DLI Queues.....	172
6.7.6 Obtaining DLI Group Resources.....	174
6.7.7 Uploading Local Files or OBS Files to a DLI Group.....	175
6.7.8 Stopping a DLI Spark Job.....	176
6.8 Using ma-cli to Copy OBS Data.....	177

1 Introduction to DevEnviron

NOTE

This document describes the DevEnviron notebook functions of the new version.

Software development is a process of reducing developer costs and improving development experience. In AI development, ModelArts is dedicated to improving AI development experience and simplifying the development process. ModelArts DevEnviron uses cloud native resources and integrates the development tool chain to provide better in-cloud AI development experience for AI development, exploration, and teaching.

ModelArts notebook for seamless in-cloud and on-premises collaboration

- In-cloud JupyterLab, local IDE, and ModelArts plug-ins for remote development and debugging, tailored to your needs
- In-cloud development environment with AI compute resources, cloud storage, and built-in AI engines
- Custom runtime environment saved as an image for training and inference

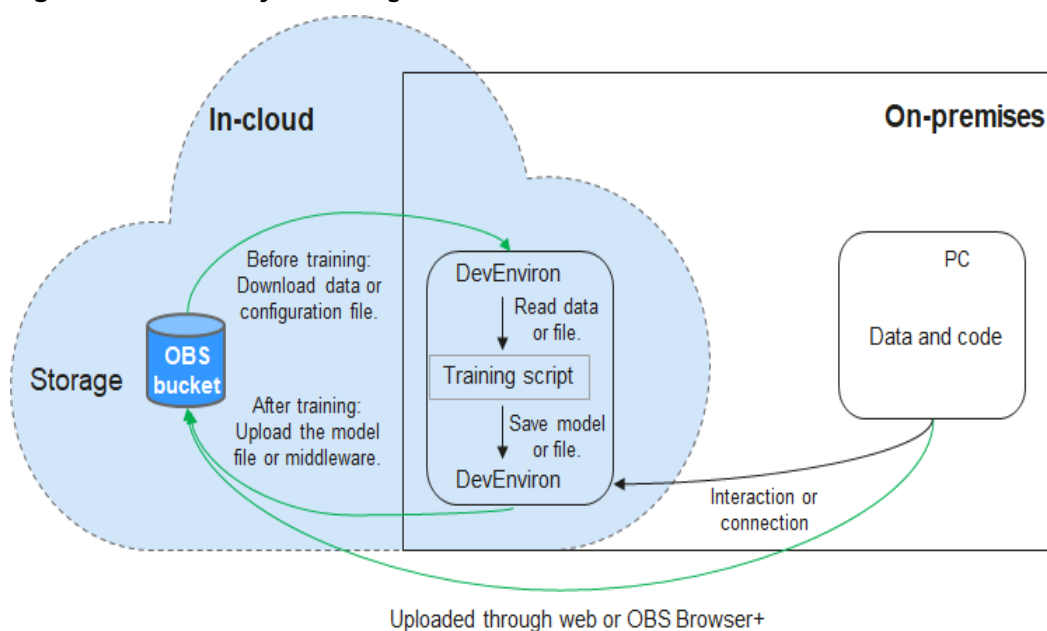
Feature 1: Remote development, allowing remote access to notebook from a local IDE

The notebook of the new version provides remote development. After enabling remote SSH, you can remotely access the ModelArts notebook development environment to debug and run code from a local IDE.

Due to limited local resources, developers using a local IDE run and debug code typically on a CPU or GPU server shared between team members. Building and maintaining the CPU or GPU server are costly.

ModelArts notebook instances are out of the box with various built-in engines and flavors for you to select. You can use a dedicated container environment. Only after simple configurations, you can remotely access the environment to run and debug code from your local IDE.

Figure 1-1 Remotely accessing notebook from a local IDE



ModelArts notebook can be regarded as an extension of a local development environment. The operations such as data reading, training, and file saving are the same as those performed in a local environment.

ModelArts notebook allows you to use in-cloud resources while with local coding habits unchanged.

A local IDE supports Visual Studio (VS) Code, PyCharm, and SSH. In addition, the PyCharm Toolkit and VS Code Toolkit plug-ins allow you to easily use cloud resources.

Feature 2: One-click image saving to save a development environment

ModelArts notebook of the new version allows you to save a running notebook instance as a custom image with one click.

When an image is saved, the installed pip dependency package is retained. In remote development through VS Code, the plug-ins installed on the server are retained.

Feature 3: Preset images that are out-of-the-box with optimized configurations and supporting mainstream AI engines

The AI engines and versions preset in each image are fixed. When creating a notebook instance, specify an AI engine and version, including the chip type.

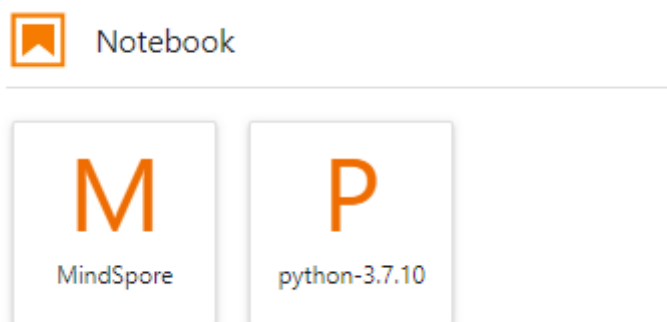
ModelArts DevEnviron provides a group of preset images, including PyTorch, TensorFlow, and MindSpore images. You can use a preset image to start your notebook instance. After the development in the instance, submit a training job without any adaptation.

The image versions preset in ModelArts are determined based on user feedback and version stability. If your development can be carried out using the versions

preset in ModelArts, for example, MindSpore 1.5, use preset images. These images have been fully verified and have many commonly-used installation packages built in. They are out-of-the-box, relieving you from configuring the environment.

The images preset in ModelArts DevEnviron include:

- Common preset packages: common AI engines such as PyTorch and MindSpore based on standard Conda, common data analysis software packages such as Pandas and Numpy, and common tool software such as CUDA and CUDNN, meeting common AI development requirements.
- 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 the AI engine 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 improving user experience.
- Remote SSH: allows you to remotely debug a notebook instance from a local PC.
- After the images preset in ModelArts DevEnviron support development, the training jobs can be executed on ModelArts.

NOTE

- To simplify operations, ModelArts notebook of the new version supports switchover between AI engines in a notebook instance.
- AI engines vary based on regions. For details about the AI engines available in a region, see the AI engines displayed on the management console.

Feature 4: JupyterLab, an online interactive development and debugging tool

ModelArts integrates open-source JupyterLab for online interactive development and debugging. You can use the notebook on the ModelArts management console to compile and debug code and train models based on the code, without concerning environment installation or configuration.

JupyterLab is an interactive development environment. It is the next-generation product of Jupyter Notebook. JupyterLab enables you to compile notebooks,

operate terminals, edit Markdown text, enable interaction, and view CSV files and images.

2 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 [JupyterLab Overview and Common Operations](#).
- 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 and [Connecting to a Notebook Instance Through VS Code Toolkit](#).

3 Managing Notebook Instances

[Creating a Notebook Instance](#)

[Accessing a Notebook Instance](#)

[Searching for, Starting, Stopping, or Deleting a Notebook Instance](#)

[Changing a Notebook Instance Image](#)

[Changing the Flavor of a Notebook Instance](#)

[Selecting Storage in DevEnviron](#)

[Dynamically Mounting an OBS Parallel File System](#)

[Dynamically Expanding EVS Disk Capacity](#)

[Modifying the SSH Configuration for a Notebook Instance](#)

[Viewing the Notebook Instances of All IAM Users Under One Tenant Account](#)

[Viewing Notebook Events](#)

[Notebook Cache Directory Alarm Reporting](#)

3.1 Creating a Notebook Instance

Before developing a model, create a notebook instance and access it for coding.

Context

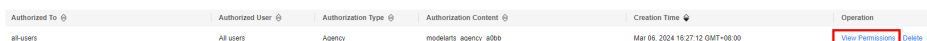
- Notebook is billed as follows:
 - A running notebook instance will be billed based on used resources. The fees vary depending on your selected resources. For details, see [Pricing Details](#). When a notebook instance is not used, stop it.
 - If you select EVS for storage when creating a notebook instance, the EVS disk will be continuously billed. Stop and delete the notebook instance if it is not required.
- 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 management console. In the navigation pane, choose **Settings** and check whether the access authorization has been configured. If not, configure access authorization. For details, see [Configuring Access Authorization](#).

Figure 3-1 Viewing agency configurations



2. Log in to the ModelArts management console. In the navigation pane on the left, choose **DevEnviron > Notebook**.
3. Click **Create** in the upper right corner. On the **Create Notebook** page, configure parameters.
 - a. Configure basic information of the notebook instance, including its name, description, and auto stop status. For details, see [Table 3-1](#).

Figure 3-2 Basic information of a notebook instance

* Name

Description

* Auto Stop

0/256

i Enable this option to specify a time for the notebook instance to automatically stop. You will not be billed after it has stopped. X

Table 3-1 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

Parameter	Description
Auto Stop	<p>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 24 hours.</p> <ul style="list-style-type: none"> • Stop as scheduled: If this option is enabled, the notebook instance automatically stops when the running duration exceeds the specified duration. <p>NOTE To protect in-progress jobs, a notebook instance does not automatically stop immediately at the auto stop time. Instead, there is a period of 2 to 5 minutes provided for you to renew the auto stop time.</p>

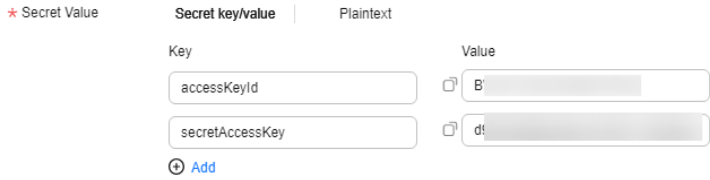
- b. Configure notebook parameters, such as the image and instance flavor. For details, see [Table 3-2](#).

Table 3-2 Notebook instance parameters

Parameter	Description
Image	<p>Public and private images are supported.</p> <ul style="list-style-type: none"> • Public images are the AI engines built in ModelArts. • Private images can be created using an instance that is created using a public image. For details, see Using Custom Images in Notebook Instances. <p>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.</p> <p>You can change an image on a stopped notebook instance.</p>
Resource Type	<p>Public and dedicated resource pools are available for you to select.</p> <p>Public resource pools are billed based on the running duration of your notebook instances.</p> <p>Select a created dedicated resource pool based on site requirements. If no dedicated resources are available, purchase one.</p>

Parameter	Description
Type	<p>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.</p>
Flavor	<p>The flavor of your notebook instance. Select a flavor based on your needs.</p> <ul style="list-style-type: none"> • CPU <ul style="list-style-type: none"> 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 <ul style="list-style-type: none"> 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	<p>The value can be EVS, SFS, OBS, or PFS. Configure this parameter based on your needs.</p> <p>NOTE OBS and PFS are whitelist functions. If you have trial requirements, submit a service ticket to apply for permissions.</p> <ul style="list-style-type: none"> ● 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 ModelArts Network. <p>NOTE For details about how to set permissions to access SFS Turbo folders, see Permissions Management.</p> <ul style="list-style-type: none"> – Scalable File Service: Select a created SFS Turbo file system. To create an SFS Turbo file system, log in to Huawei Cloud. – 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. <ul style="list-style-type: none"> ● 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
	<p>Figure 3-3 Configuring the secret values</p>  <p>All storage paths of EVS and SFS are mounted to the /home/ma-user/work directory. All read and write operations on files in the notebook instance are stored in this directory, not in OBS.</p> <p>You can add a data storage path during the runtime of a notebook instance by referring to Dynamically Mounting an OBS Parallel File System.</p> <p>The data is retained in /home/ma-user/work, even if the notebook instance is stopped or restarted.</p> <p>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.</p>
Extended Storage	<p>NOTE</p> <p>This parameter is a whitelist function. If you have trial requirements, submit a service ticket to apply for permissions.</p> <p>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.</p> <p>Constraints:</p> <ul style="list-style-type: none"> • 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/ <p>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 on the right.</p>

Parameter	Description
Remote SSH	<ul style="list-style-type: none"> 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. <p>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.</p>
Key Pair	<p>Set a key pair after remote SSH is enabled. Select an existing key pair.</p> <p>Alternatively, click Create on the right of the text box to create one on the DEW console. To do so, choose Key Pair Service > Private Key Pairs and click Create Key Pair.</p> <p>After a notebook instance is created, you can change the key pair on the instance details page.</p> <p>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.</p>
Whitelist	<p>Set a whitelist after remote SSH is enabled. This parameter is optional.</p> <p>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.</p> <p>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.</p> <p>After a notebook instance is created, you can change the whitelist IP addresses on the instance details page.</p>

- c. (Optional) Add tags to the notebook instance. Enter a tag key and value and click **Add**.

Table 3-3 Adding a tag

Parameter	Description
Tags	<p>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.</p> <p>For details about how to use tags, see How Does ModelArts Use Tags to Manage Resources by Group?</p> <p>After adding a tag, you can view, modify, or delete the tag on the notebook instance details page.</p>

 **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.

4. Click **Next**.
5. After confirming the parameter settings, click **Submit**.
Switch to the notebook instance list. The notebook instance is being created. It will take several minutes when its status changes to **Running**. Then, the notebook instance is created.
6. 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.

On the **Storage** tab page, 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](#).

3.2 Accessing a Notebook Instance

Access a notebook instance in the **Running** state for coding.

The methods of accessing notebook instances vary depending on the AI engine based on which the instance was created.

- Remote access: Use PyCharm, VS Code, or SSH in the local IDE. For details, see [Connecting to a Notebook Instance Through VS Code Toolkit](#) and .
- Online access: Use JupyterLab. For details, see [JupyterLab Overview and Common Operations](#).

Create an instance and mount the persistent storage to `/home/ma-user/work`.

```
sh-4.4$pwd
/home/ma-user
sh-4.4$cd work/
sh-4.4$pwd
/home/ma-user/work
sh-4.4$
```

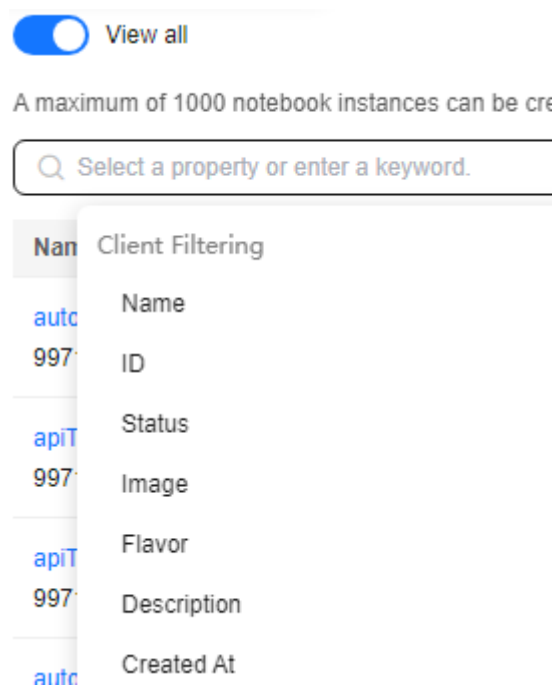
The data stored in only the `work` directory is retained after the instance is stopped or restarted. When you use a development environment, store the data for persistence in `/home/ma-user/work`.

3.3 Searching for, Starting, Stopping, or Deleting 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. Click the search box and select one or more search criteria.

Figure 3-4 Searching for an Instance

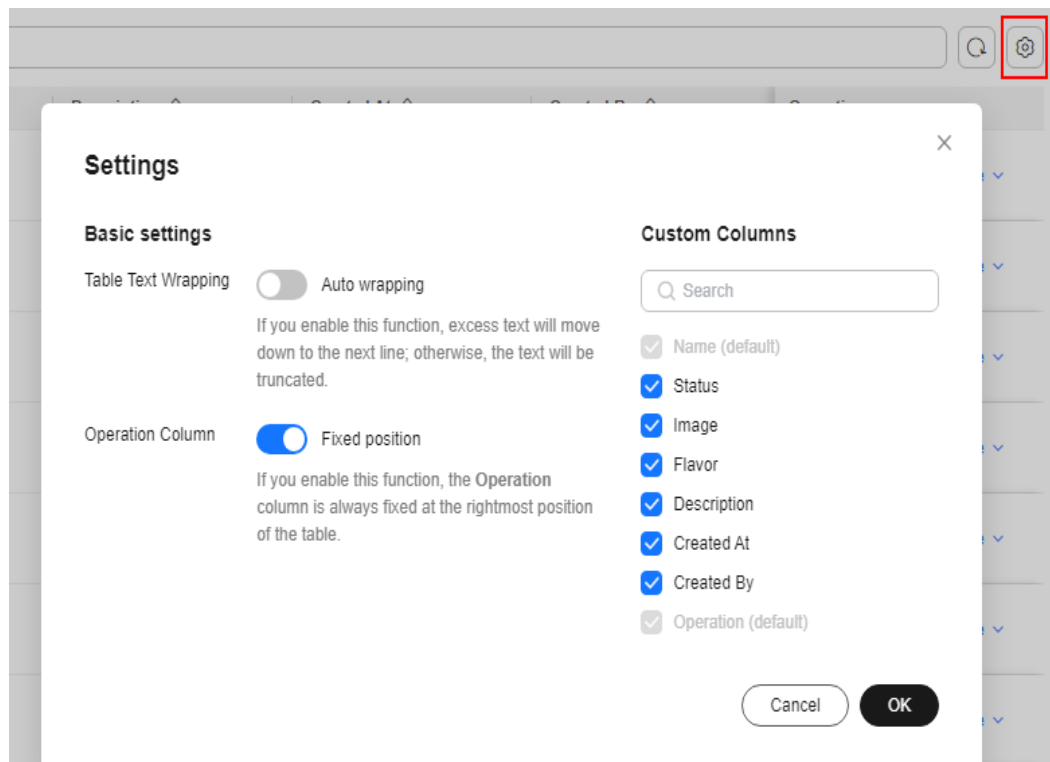


- Enable **View all** to check all notebook instances created by all sub-users in the IAM project.
- Select search criteria, such as name, ID, status, image, flavor, description, and creation time.

Customizing Table Columns

Click the settings button to customize the columns to be displayed in the table.

Figure 3-5 Settings



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 management console. In the navigation pane on the left, choose **DevEnviron > 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:

- The data stored only in `/home/ma-user/work` is retained. For example, the external dependency packages installed in other directories in the development environment will be deleted.
 - The notebook instance will no longer be billed. However, if the instance is attached with an EVS disk, the storage space will still be billed.
-

Deleting an Instance

Delete the notebook instances that are not needed.

1. Log in to the ModelArts management console. In the navigation pane on the left, choose **DevEnviron > 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.

3.4 Changing a Notebook Instance Image

ModelArts allows you to change images on a notebook instance to flexibly adjust its AI engine.

Constraints

The target notebook instance is stopped.

Procedure

1. Log in to the ModelArts management console. In the left navigation pane, choose **DevEnviron > Notebook**.
2. In the notebook list, click **More** in the **Operation** column of the target notebook instance and select **Change Image**.
3. In the **Change Image** dialog box, select a new image and click **OK**. After the modification, you can view the new image on the notebook list page.

3.5 Changing the Flavor of a Notebook Instance

ModelArts allows you to change the node flavor for a notebook instance.

Constraints

Specifications of a notebook instance can be modified only when the notebook instance is in the **Stopped**, **Running**, or **Startup failed** state.

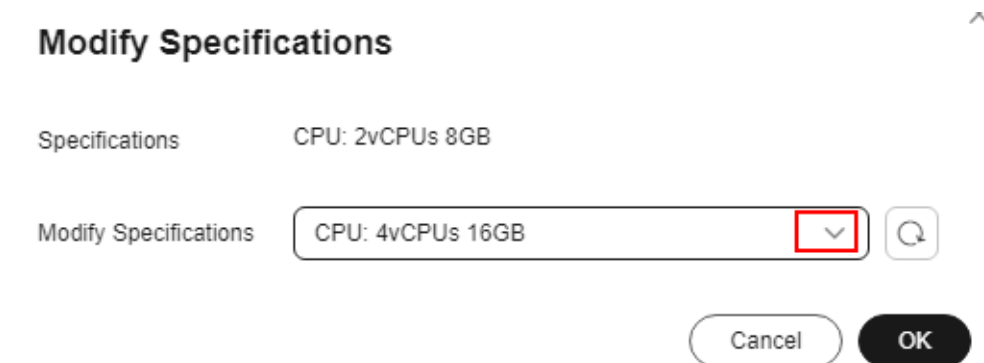
Procedure

1. Log in to the ModelArts management console. In the navigation pane on the left, choose **DevEnviron > Notebook**.
2. In the notebook instance list, locate the row that contains the target notebook instance and choose **More > Modify Specifications** in the **Operation** column. In the **Modify Specifications** dialog box that appears, select the required flavor.

Figure 3-6 Modify Specifications



Figure 3-7 Selecting a flavor



3.6 Selecting Storage in DevEnviron

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.

NOTE

Only OBS parallel file systems (PFS) and object storage in the same region can be mounted.

Table 3-4 In-cloud storage application scenarios

Storage	Application Scenario	Advantage	Disadvantage
EVS	Data and algorithm exploration only in the development environment.	<p>Block storage SSDs feature better overall I/O performance than NFS. The storage capacity can be dynamically expanded to up to 4096 GB.</p> <p>As persistent storage, EVS disks are mounted to <code>/home/ma-user/work</code>. The data in this directory is retained after the instance is stopped. The storage capacity can be expanded online based on demand.</p>	This type of storage can only be used in a single development environment.

Storage	Application Scenario	Advantage	Disadvantage
PFS	<p>NOTE PFS is a whitelist function. To use this function, contact Huawei technical support.</p> <p>PFS buckets mounted as persistent storage for AI development and exploration.</p> <ul style="list-style-type: none"> - Storage for datasets. Datasets are directly mounted to notebooks for browsing and data processing and can be directly used during training. For details, see How Do I Upload Data to OBS? <p>After the instance is running, the OBS parallel file system that carries the datasets is dynamically mounted to notebooks. For details, see Dynamically Mounting an OBS Parallel File System.</p> <p>2. Storage for code. After debugging on a notebook instance, specify the OBS path as the code path for starting training, facilitating temporary modification.</p> <ul style="list-style-type: none"> - 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 	<p>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.</p> <p>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.</p>	<p>Due to average performance in frequent read and write of small files, PFS storage is not suitable for large model training or file decompression.</p> <p>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.</p>

Storage	Application Scenario	Advantage	Disadvantage
	<p>real time. This is especially suitable for analyzing the output of jobs trained using TensorBoard or notebook.</p>		
OBS	<p>NOTE OBS is a whitelist function. To use this function, contact Huawei technical support.</p> <p>When uploading or downloading a large amount of data in the development environment, you can use OBS buckets to transfer data.</p>	<p>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.</p>	<p>The object storage semantics is different from the Posix semantics and needs to be further understood.</p>
SFS	<p>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.</p>	<p>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.</p>	<p>The performance of the SFS storage is not as good as that of the EVS storage.</p>

Storage	Application Scenario	Advantage	Disadvantage
Local storage	First choice for heavy-duty training jobs.	<p>High-performance SSDs for the target VM or BMS, featuring high file I/O throughput. For heavy-duty training jobs, store data in the target directory and then start training.</p> <p>By default, the storage is mounted to the <code>/cache</code> directory. For details about the available space of the <code>/cache</code> directory, see What Are Sizes of the /cache Directories for Different Notebook Specifications in DevEnviron?.</p>	The storage lifecycle is associated with the container lifecycle. Data needs to be downloaded each time the training job starts.

Using the Storage

- 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](#).

3.7 Dynamically Mounting an OBS Parallel File System

Overview

Parallel File System (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 PFS for storage.

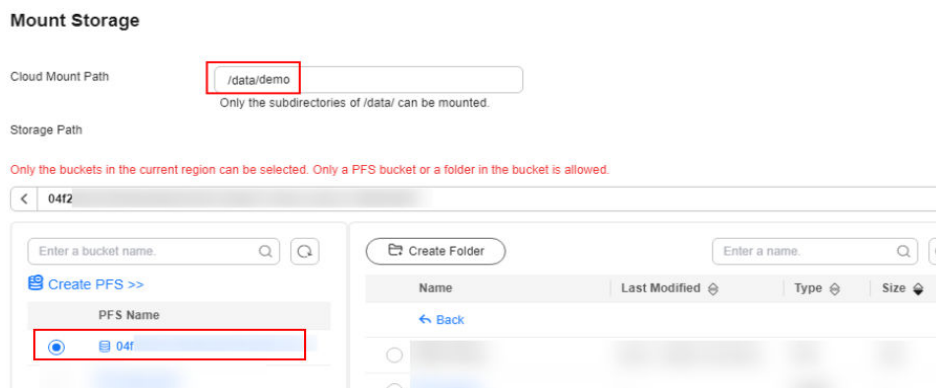
The ModelArts notebook of the new version supports only the mounting of an OBS parallel file system to **/data/** of a notebook container.

Procedure

Method 1: Through the ModelArts management console

1. Log in to the ModelArts management console. In the left navigation pane, choose **DevEnviron > 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 3-8 Dynamically mounting an OBS parallel file system



3. View the mounting result on the notebook instance details page.

Figure 3-9 Successful mounting

Type	Status	Storage Path	Cloud Mount Path	Operation
Parallel File System	Mounted	obs://	/data/demo/	Unmount Storage

3.8 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.

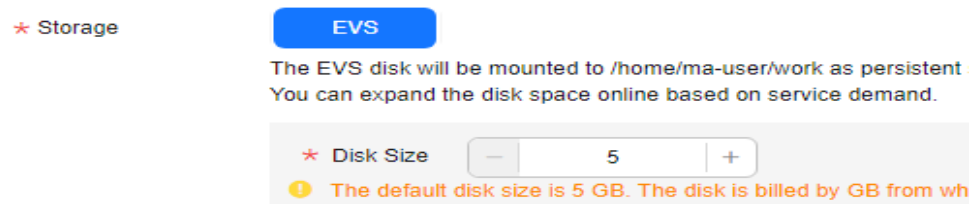
Application Scenarios

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 3-10 Selecting EVS when creating a notebook instance

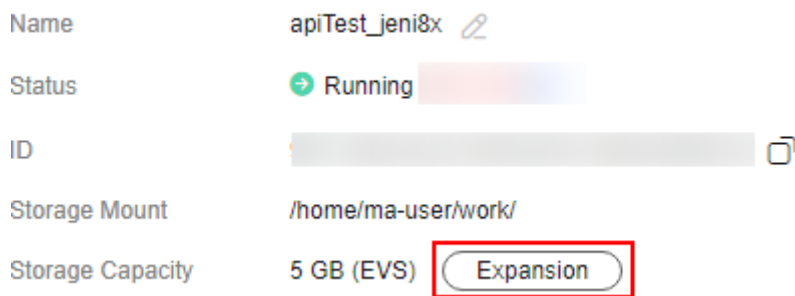


- Up to 100 GB can be expanded at a time. Additionally, the total capacity after expansion cannot exceed 4096 GB.
- If the original capacity of an EVS disk is 4096 GB, the disk capacity cannot be expanded.
- After the instance is stopped, the expanded capacity still takes effect. The billing is based 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 left navigation pane, choose **DevEnviron** > **Notebook**.
2. Click the name of a running notebook instance. On the instance details page, click **Expansion**.

Figure 3-11 Instance details page



3. 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 3-12 Capacity expansion

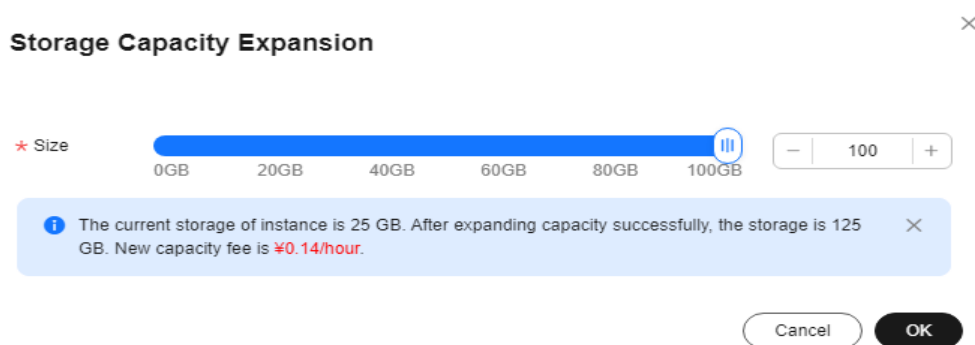
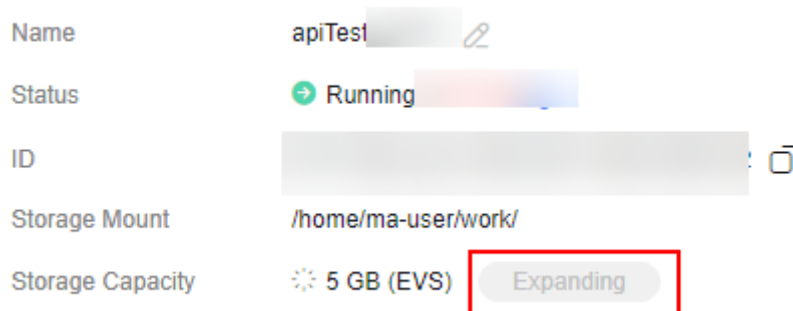


Figure 3-13 Expanding



3.9 Modifying the SSH Configuration for a Notebook Instance

ModelArts allows you to modify the SSH configuration for notebook instances.

If a notebook instance is created with remote SSH disabled, you can enable remote SSH on the notebook details page.

During the creation of a notebook instance, if you set a whitelist for remotely accessing it, you can change the IP addresses in the whitelist on the notebook instance details page. You can also change the key pair.

Constraints

The target notebook instance must be stopped.

Changing the Key Pair and Remote Connection IP Address

1. Log in to the ModelArts management console. In the navigation pane on the left, choose **DevEnviron** > **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 3-14](#). After the SSH configuration is updated, the remote SSH function cannot be disabled.

For remote SSH enabled by default in the selected image, see [Figure 3-15](#).

Figure 3-14 Update SSH Configuration

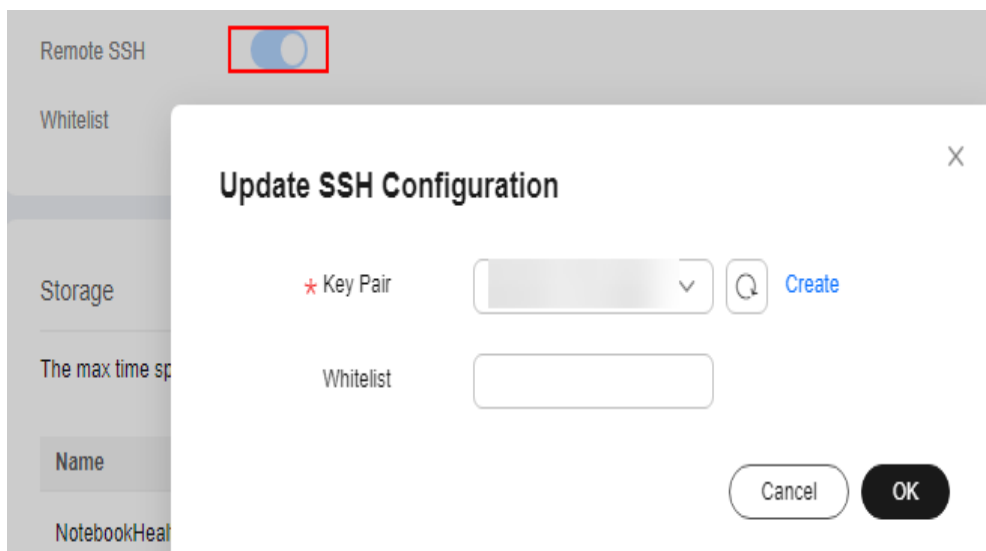
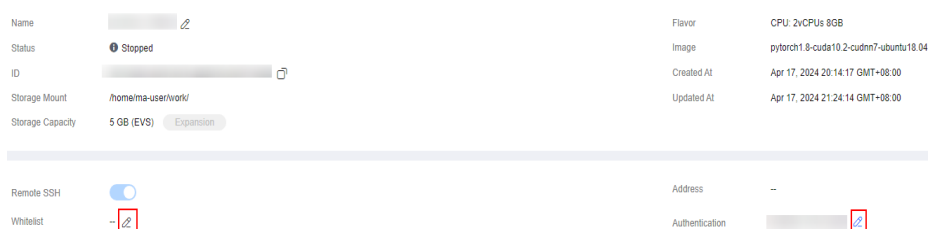



Figure 3-15 Changing the whitelist and key pair

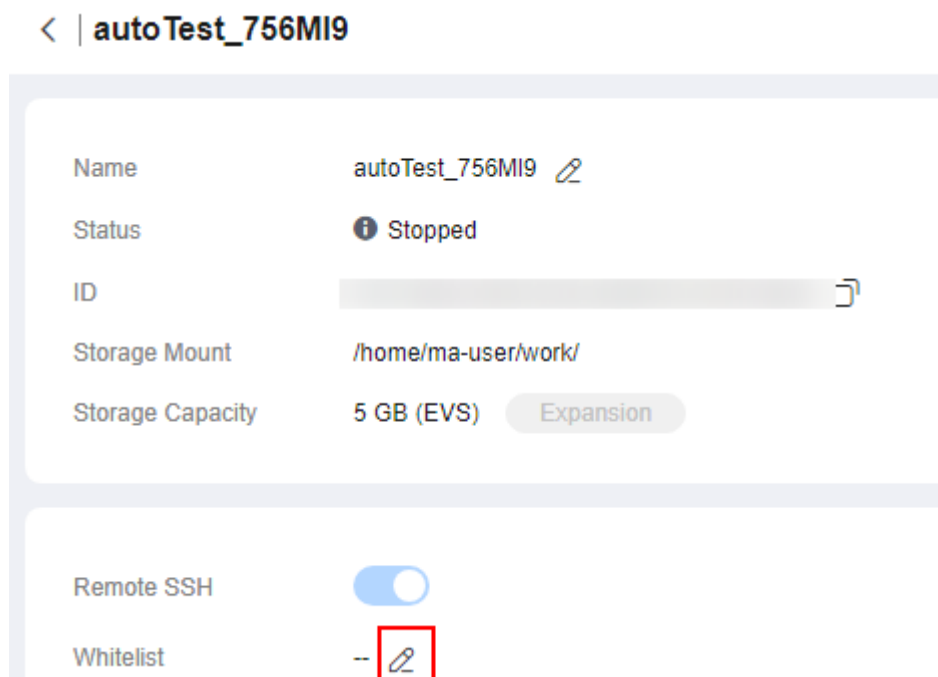


- Click  and choose an existing key pair, or click **Create** to create a new key pair.
- For details about how to configure a whitelist, see [Setting an IP Address for Remotely Accessing a Notebook Instance](#). After you change the IP

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.

Setting an IP Address for Remotely Accessing a Notebook Instance

Figure 3-16 Setting an IP address for remotely accessing a notebook instance



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.

3.10 Viewing the Notebook Instances of All IAM Users Under One Tenant Account

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.

NOTE

Users granted with these permissions can also access OBS and SWR of all users in the current IAM project.

Assigning the Required Permissions

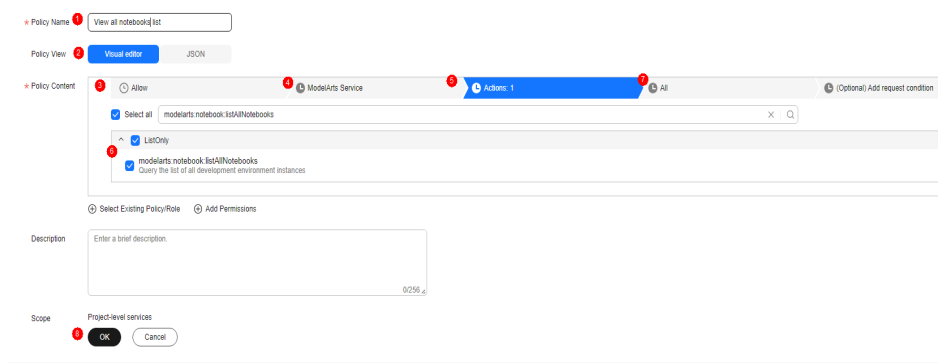
1. Log in to the ModelArts management console as a tenant user, hover the cursor over your username in the upper right corner, and choose **Identity and Access Management** from the drop-down list to switch to the IAM management console.

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 3-17](#).

- **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 3-17 Creating a custom policy



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.6789** will be reported. For details about how to update a key pair, see [Modifying the SSH Configuration for a Notebook Instance](#).

Error message: ModelArts.6789: Failed to use SSH key pair KeyPair-xxx. Update the key pair and try again later.

3.11 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.

Figure 3-18 Viewing notebook instance events and configuring automatic refresh

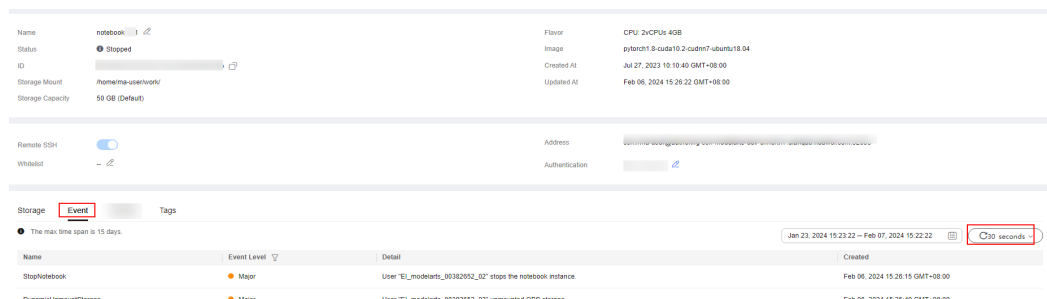


Table 3-5 Events during instance creation

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

Table 3-6 Events during instance startup

Event Name	Description	Severity
Scheduled	The instance has been scheduled.	Warning
PullingImage	The image is being pulled.	Warning
PulledImage	The image has been pulled.	Warning

Event Name	Description	Severity
NotebookHealthy	The instance is running and healthy.	Major
RunHookScript	Running a custom script	Warning
StartNotebookFailed	Starting the instance failed.	Critical
PullImageFailed	Pulling the image failed.	Critical
CreateKernelFailed	Creating a Jupyter kernel failed because the conda command is unavailable. (The conda environments are not being detected and added as Jupyter kernels. Ensure that <code>{conda_env}</code> is available and the command <code>{conda_cmd}</code> env list can be run properly.)	Major
	Creating a Jupyter kernel failed due to permission issues. (Kernels are not showing up in Jupyter Notebook due to permission issues. Ensure that the uid <code>{ma_uid}</code> has write permissions on <code>{conda_path}</code> .)	Major
ConfigurationError	Configuring the ModelArts SDK and CLI paths in the conda environment failed due to unavailable conda command. (The ModelArts SDK and CLI are unavailable in the conda environments due to conda environment issues. Ensure that <code>{conda_env}</code> is available and the command <code>{conda_cmd}</code> env list can be run properly.)	Major
	Configuring the ModelArts SDK and CLI paths in the conda environment failed due to permission issues. (The ModelArts SDK and CLI are unavailable in the conda environments due to conda environment issues. Ensure that the uid <code>{ma_uid}</code> has write permissions on <code>{conda_path}</code> .)	Major

Table 3-7 Events during instance stopping

Event	Description	Severity
StopNotebook	The instance has been stopped.	Major

Event	Description	Severity
StopNotebookResourceIdle	The notebook instance will automatically stop or has automatically stopped because resources are idle.	Major

Table 3-8 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
	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

Event	Description	Severity
	Updating the storage size failed due to an internal error. (Updating the EVS disk size failed. The O&M personnel are handling the fault.)	Critical

Table 3-9 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
	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 personnel are handling the fault.)	Critical
	Saving the image failed due to SWR service issues.	Critical

Table 3-10 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 3-11 Events for dynamic OBS mounting

Event	Description	Severity
DynamicMountStorage	The OBS storage is mounted.	Major
DynamicUnmountStorage	The OBS storage is unmounted.	Major

Table 3-12 Events triggered on the user side

Event	Description	Severity
RefreshCredentialsFailed	Authentication failed.	Critical

3.12 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

Enter a rule name.

Description

Enter a 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 3-19 Select Monitored Object

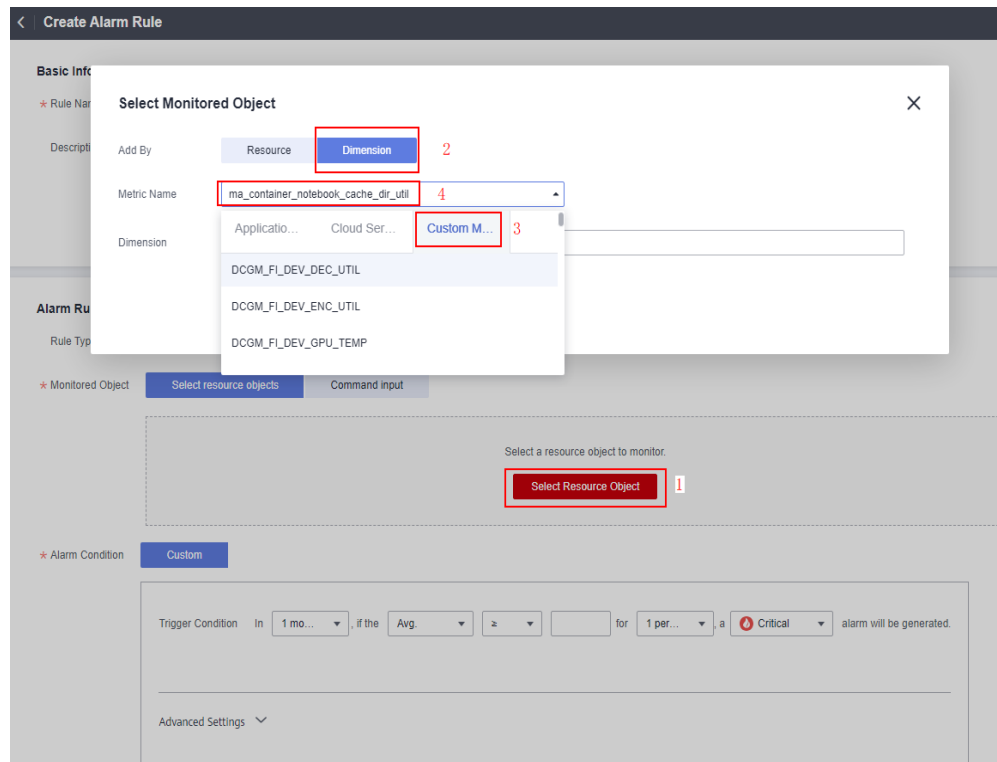


Figure 3-20 Configuring statistics method

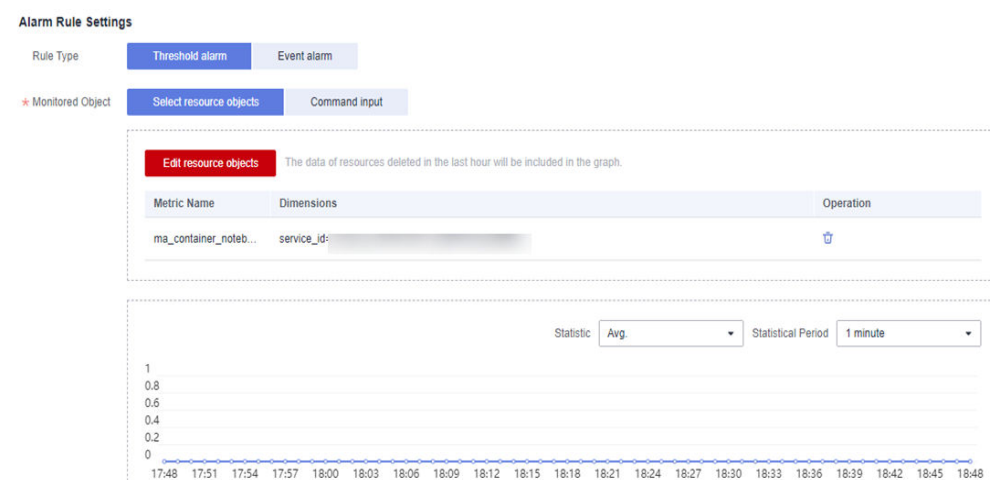
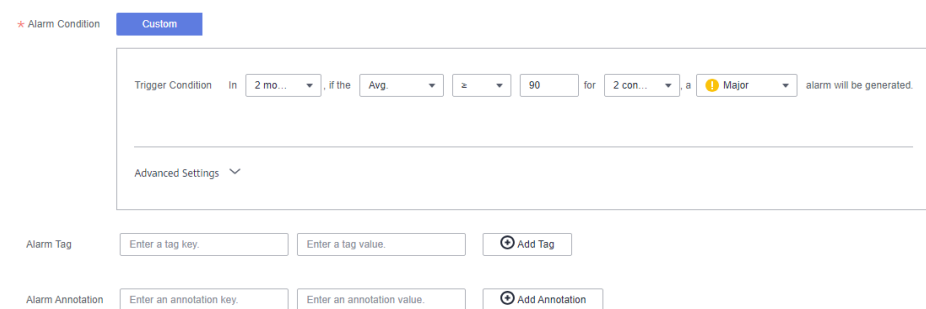


Figure 3-21 Configuring alarm conditions



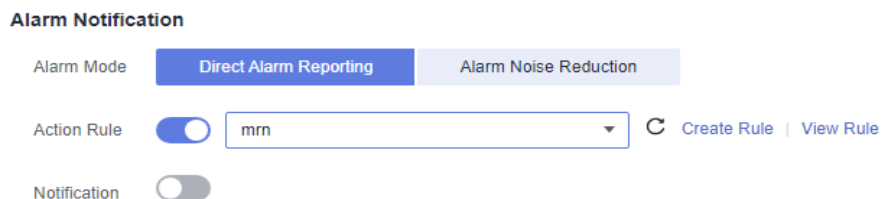
5. Configure alarm notifications and click **Create Now**.

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 3-22 Configuring alarm notifications

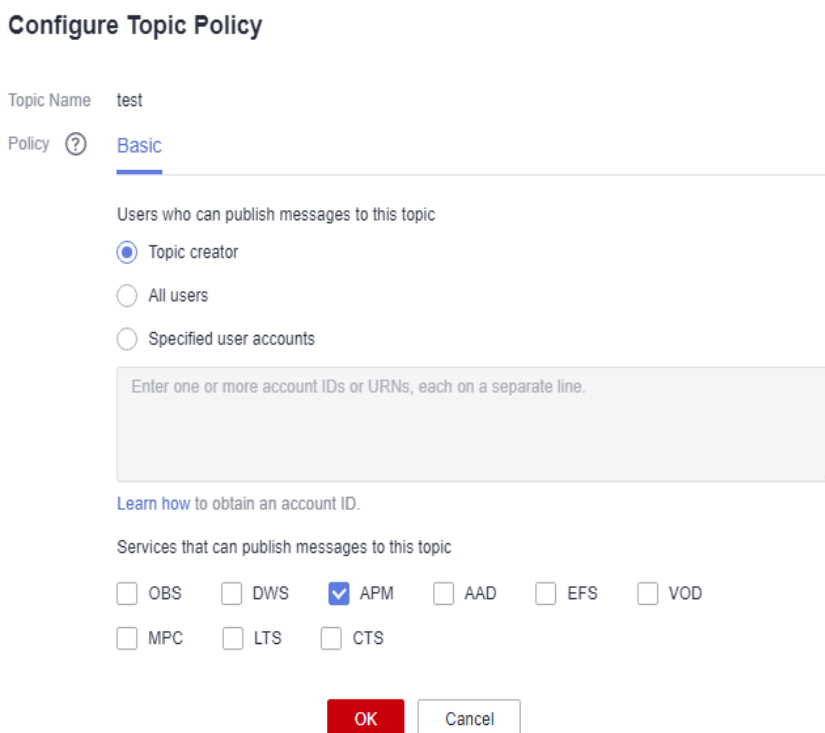


Create a topic in SMN to configure alarm notification rules.

– **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 3-23 Configure Topic Policy



- 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

* Protocol

* Endpoint ?	Endpoints	Description
	<input type="text"/>	<input type="text"/>

[+ Add Endpoint](#)
[Batch Add Endpoints](#)

A record is displayed in the subscription list, but the record is in the **Unconfirmed** state.

<input type="checkbox"/>	Subscription URN	Protocol	Endpoint	Request Header	Description	Topic Name	Status	Operation
<input type="checkbox"/>	um.smn.ap-southeast-1.1b0...	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 3-24 Create Alarm Action Rule

Create Alarm Action Rule

* 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
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

* Topic [C](#)
[If you do not see a topic you like, create one on the SMN console.](#)

* Message Template [C](#) [Create Template](#) | [View Template](#)

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.

4 JupyterLab

Operation Process in JupyterLab

JupyterLab Overview and Common Operations

Code Parametrization Plug-in

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.

Using ModelArts SDK

Using the Git Plug-in

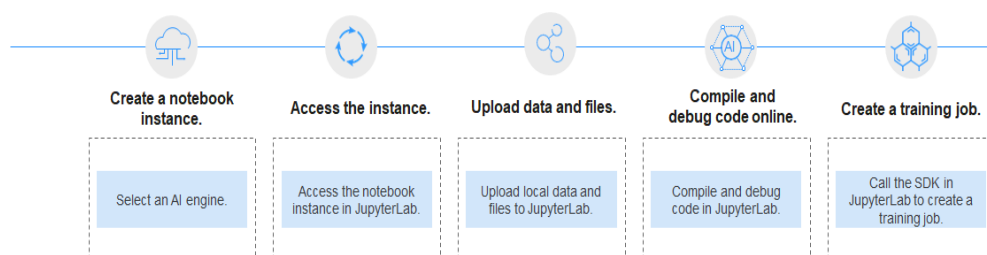
Visualized Model Training

Uploading and Downloading Data in Notebook

4.1 Operation Process in JupyterLab

ModelArts allows you to access notebook instances online using JupyterLab and develop AI models based on the PyTorch, TensorFlow, or MindSpore engines. The following figure shows the operation process.

Figure 4-1 Using JupyterLab to develop and debug code online



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](#).
2. Use JupyterLab to access the notebook instance. For details, see [Accessing JupyterLab](#).

3. Upload training data and code files to JupyterLab. For details, see [Uploading Files from a Local Path to JupyterLab](#).
4. Compile and debug code in JupyterLab. For details, see [JupyterLab Overview and Common Operations](#).
5. In JupyterLab, call the ModelArts SDK to create a training job for in-cloud training.
For details, see [Creating a Training Job](#).

4.2 JupyterLab Overview and Common Operations

JupyterLab is the next-generation web-based interactive development environment of Jupyter Notebook, 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. It has the same components as Jupyter Notebook, but offering more flexible and powerful functions.

Accessing JupyterLab

To access JupyterLab from a running notebook instance, perform the following operations:

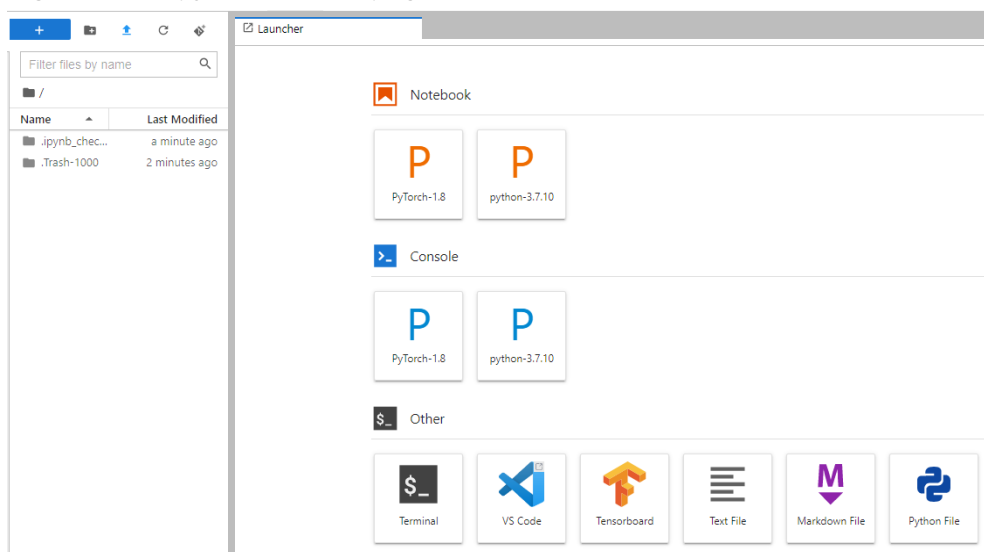
1. Log in to the ModelArts management console. In the navigation pane on the left, choose **DevEnviron > Notebook**.
2. Click **Open** in the **Operation** column of a running notebook instance to access JupyterLab.

Figure 4-2 Accessing a notebook instance

Name	Status	Image	Flavor	Description	Created At	Created By	Operation
jupyterlab-1778 40769746-746c-436c-87b-6729a2a77300	Running (58 minutes left)	spar3.1-1-ubuntu/18.04	CPU: 2VCPU; 8GB	--	Mar 20, 2024 11:16:32 GM...		Open Start More

3. The **Launcher** page is automatically displayed. Perform required operations. For details, see [JupyterLab Documentation](#).

Figure 4-3 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 4-3](#) shows an example only. Obtain the notebook and console kernels and versions on the management console.

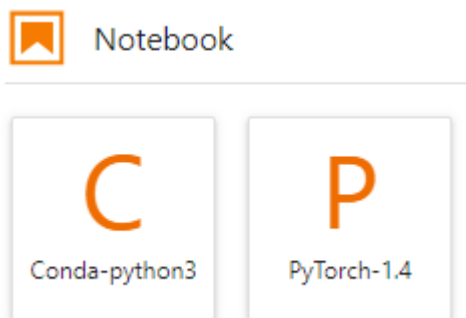
- **Notebook:** Select a kernel for running notebook, for example, TensorFlow or Python.
- **Console:** Call the terminal for command control.
- **Other:** Edit other files.

Creating an IPYNB File in JupyterLab

On the JupyterLab homepage, click a proper AI engine in the **Notebook** area to create an IPYNB file.

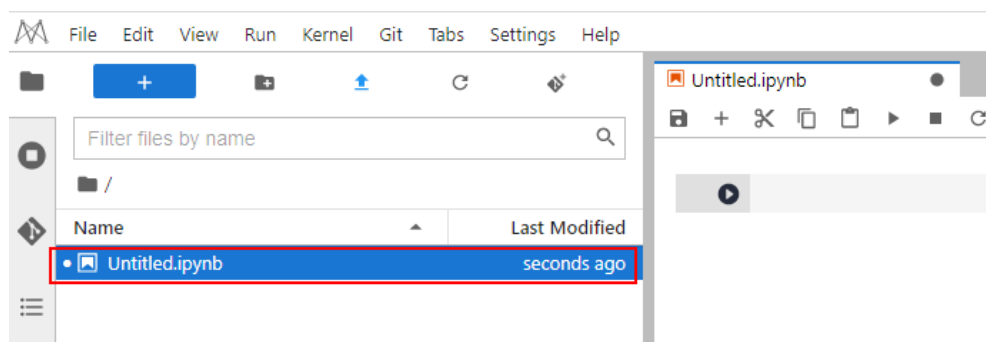
The AI engines supported by each notebook instance vary depending on the runtime environment. The following figure is only an example. Select an AI engine based on site requirements.

Figure 4-4 Selecting an AI engine and creating IPYNB file



The created IPYNB file is displayed in the navigation pane on the left.

Figure 4-5 Created IPYNB file



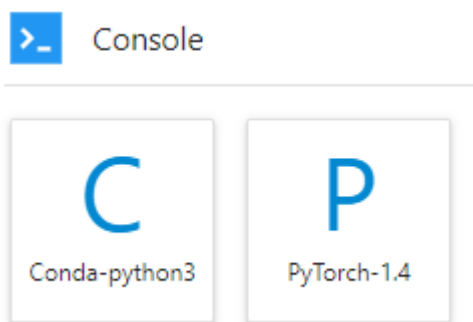
Creating a Notebook File and Accessing the Console

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.

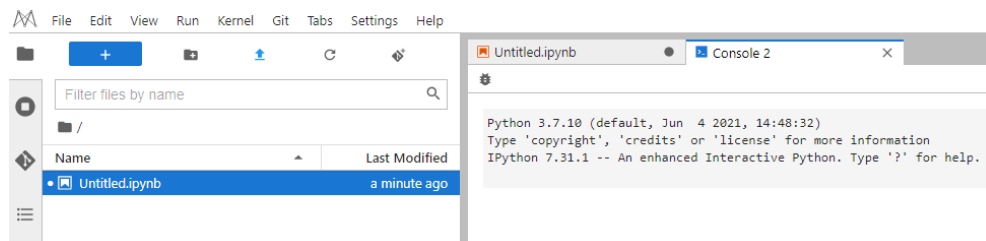
The AI engines supported by each notebook instance vary depending on the runtime environment. The following figure is only an example. Select an AI engine based on site requirements.

Figure 4-6 Selecting an AI engine and creating a console



After the file is created, the console page is displayed.

Figure 4-7 Creating a notebook file (console)

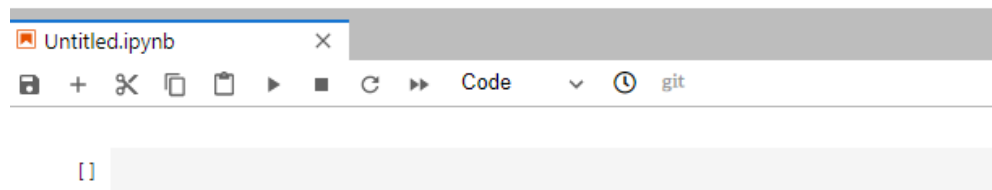
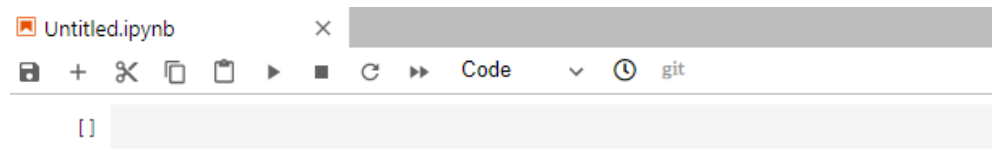


Editing a File in JupyterLab

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.

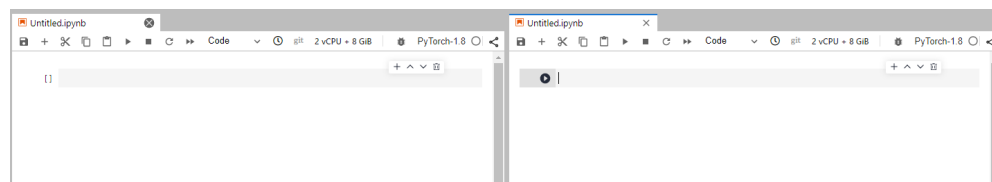
Figure 4-8 Customized display of multiple files



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 4-9 Multiple views of a file



Before coding in the code area of an IPYNB file in JupyterLab, add an exclamation mark (!) before the code.

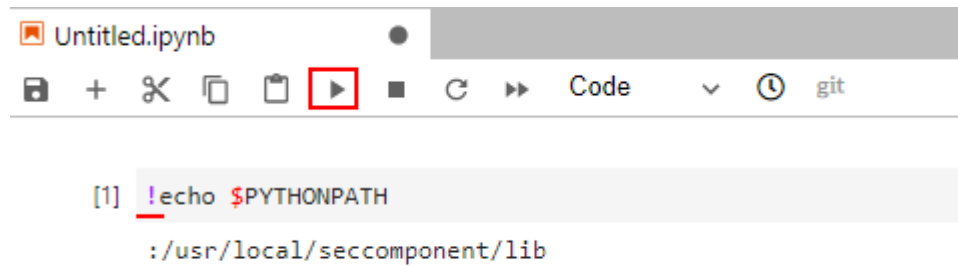
For example, install an external library Shapely.

```
!pip install Shapely
```

For example, obtain PythonPath.

```
!echo $PYTHONPATH
```

Figure 4-10 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 4-11 Remaining duration

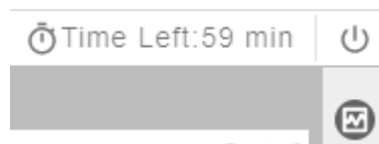
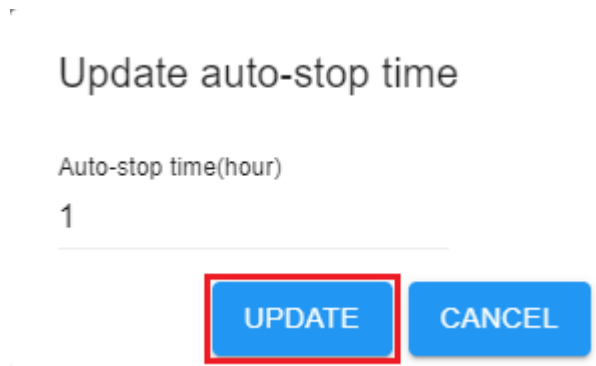


Figure 4-12 Renewing an instance



Common JupyterLab Buttons and Plug-ins

Figure 4-13 Common JupyterLab buttons and plug-ins

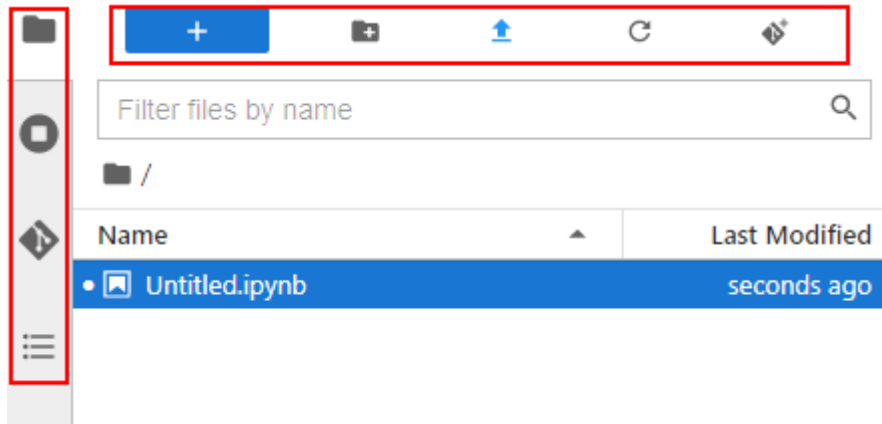


Table 4-1 JupyterLab buttons

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 4-2 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 4-14 Buttons in the navigation bar




Table 4-3 Buttons in the navigation bar









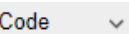


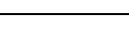
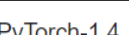


Button	Description
File	Actions related to files and directories, such as creating, closing, or saving notebooks.
Edit	Actions related to editing documents and other activities in the IPYNB file, such as undoing, redoing, or cutting cells.
View	Actions that alter the appearance of JupyterLab, such as 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 4-15 Buttons in the menu bar of an IPYNB file



Table 4-4 Buttons in the menu bar of an IPYNB file

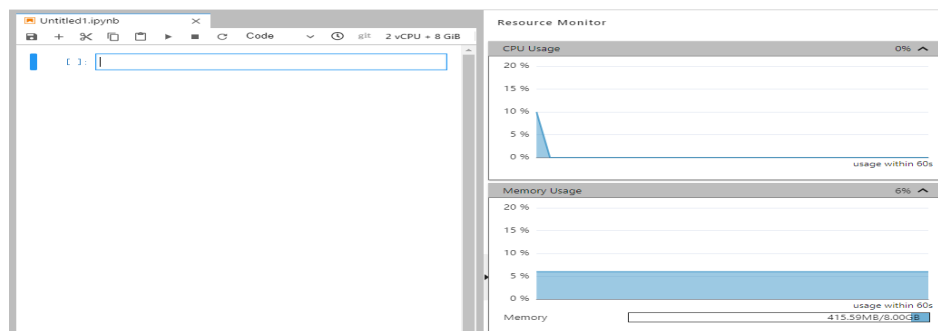
Button	Description
	Save a file.

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.

Monitoring Resources

To obtain resource usage, select **Resource Monitor** in the right pane. The CPU usage and memory usage can be viewed.

Figure 4-16 Resource usage



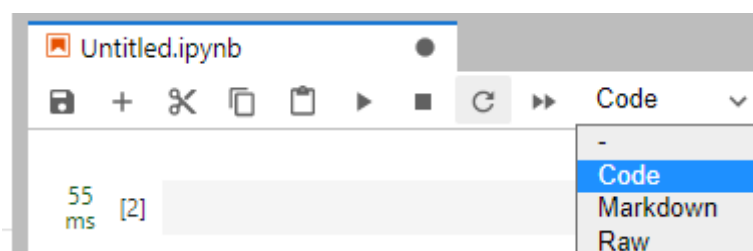
4.3 Code Parametrization Plug-in

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.

Use Guide

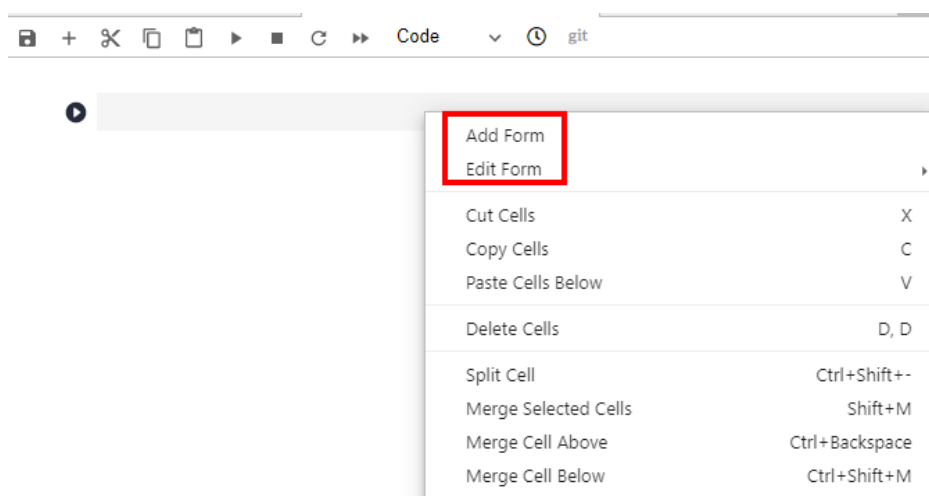
- The **Add Form** and **Edit Form** buttons are available only to the shortcut menu of code cells.

Figure 4-17 Viewing a code cell



- After opening new code, add a form before editing it.

Figure 4-18 Shortcut menu of code cells



Add Form

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 4-19 Two edit areas



Edit Form

If you click **Edit Form**, four sub-options will be displayed: **Add new form field**, **Hide code**, **Hide form**, and **Show All**.

- You can set the form field type to **dropdown**, **input**, and **slider**. See [Figure 4-20](#). 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. See [Figure 4-21](#).

Figure 4-20 Form style of dropdown, input, and slider

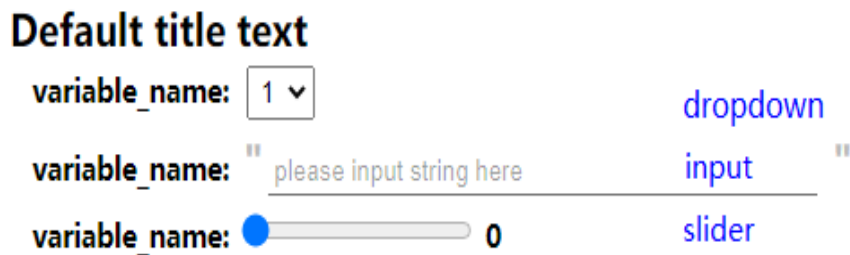


Figure 4-21 Creating a dropdown form

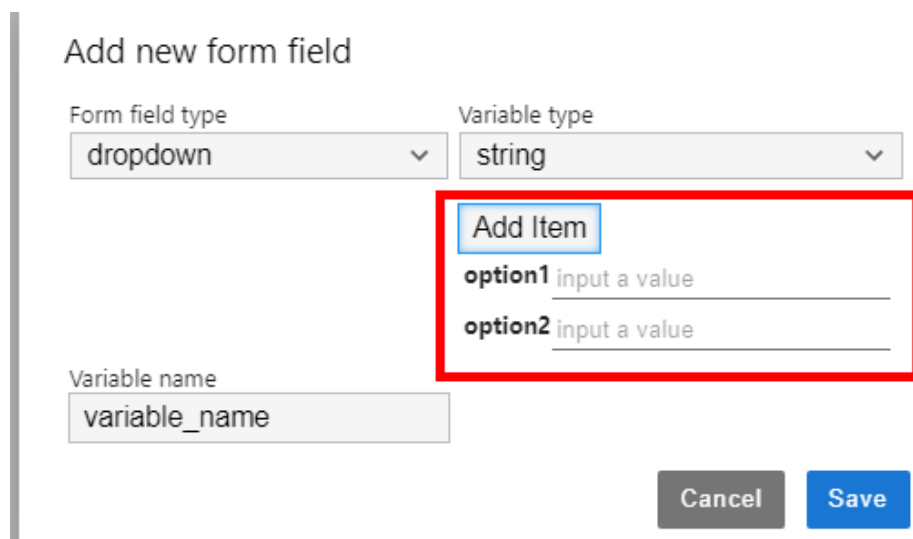


Figure 4-22 Deleting a form



- 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.
- If you click **Hide code**, the code area will be hidden.
- If you click **Hide form**, the form area will be hidden.
- If you click **Show All**, both the code and form areas will be displayed.

4.4 Using ModelArts SDK

Notebook instances allow you to use ModelArts SDK to manage OBS, training jobs, models, and real-time services.

Your notebook instances have automatically obtained your AK/SK for authentication and the region. Therefore, SDK sessions are automatically authenticated.

Example Code

- Create a training job.

```
from modelarts.session import Session
from modelarts.estimator import Estimator
session = Session()
estimator = Estimator(
    modelarts_session=session,
    framework_type='PyTorch', # AI engine name
    framework_version='PyTorch-1.0.0-python3.6', # AI engine version
    code_dir='/obs-bucket-name/src/', # Training script directory
    boot_file='/obs-bucket-name/src/pytorch_sentiment.py', # Training boot script
    directory
        log_url='/obs-bucket-name/log/', # Training log directory
        hyperparameters=[
            {"label": "classes",
             "value": "10"},
            {"label": "lr",
             "value": "0.001"}
        ],
        output_path='/obs-bucket-name/output/', # Training output directory
        train_instance_type='modelarts.vm.gpu.p100', # Training environment
    specifications
        train_instance_count=1, # Number of training nodes
        job_description='pytorch-sentiment with ModelArts SDK' # Training job description
)
job_instance = estimator.fit(inputs='/obs-bucket-name/data/train/', wait=False,
                             job_name='my_training_job')
```
- Obtain a model list.

```
from modelarts.session import Session
from modelarts.model import Model
```

```
session = Session()
model_list_resp = Model.get_model_list(session, model_status="published", model_name="digit",
order="desc")
```

- Obtain service details.

```
from modelarts.session import Session
from modelarts.model import Predictor
session = Session()
predictor_instance = Predictor(session, service_id="input your service_id")
predictor_info_resp = predictor_instance.get_service_info()
```

4.5 Using the Git Plug-in

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

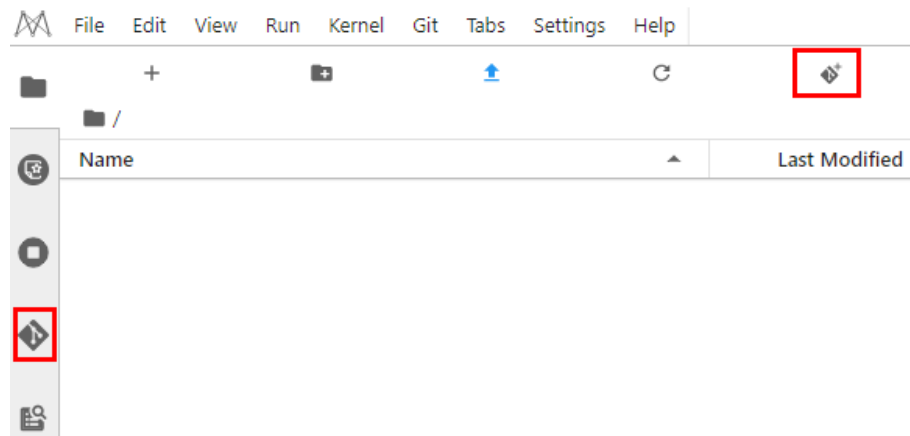
The notebook instance is running.

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 4-23](#) shows the Git plug-in of JupyterLab.

Figure 4-23 Git plug-in



Cloning a GitHub Open-Source Code Repository

Access a GitHub open-source code repository at <https://github.com/jupyterlab/extension-examplesitHub>. 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

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

username

password / personal access to

Cancel OK

To obtain a personal access token, perform the following operations:

1. Log in to [GitHub](#) and open the configuration page.
2. Click **Developer settings**.
3. Choose **Personal access tokens > Generate new token**.
4. Verify the account.
5. Describe the token, select permissions to access the private repository, and click **Generate token** to generate a token.
6. 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 4-24 Cloning a GitHub private code repository (only authorization using a personal access token is supported)

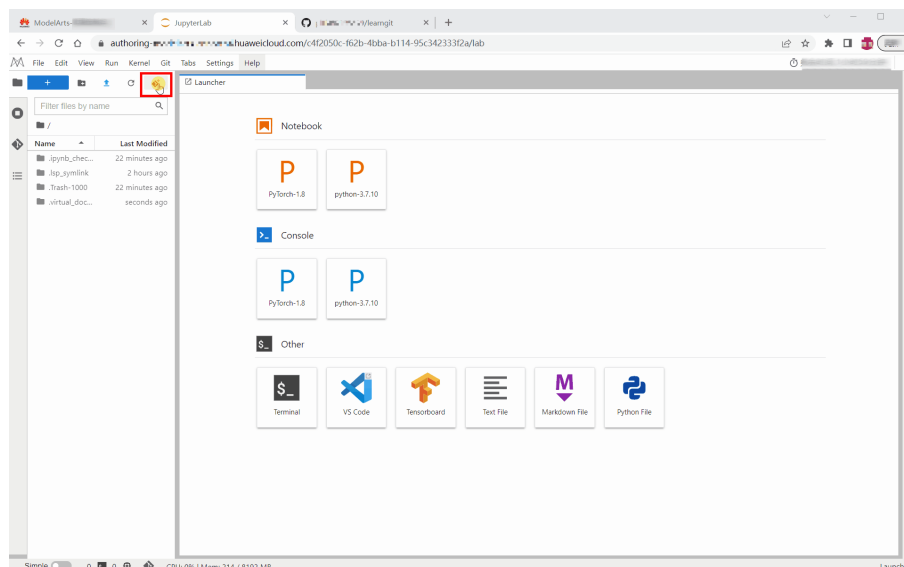
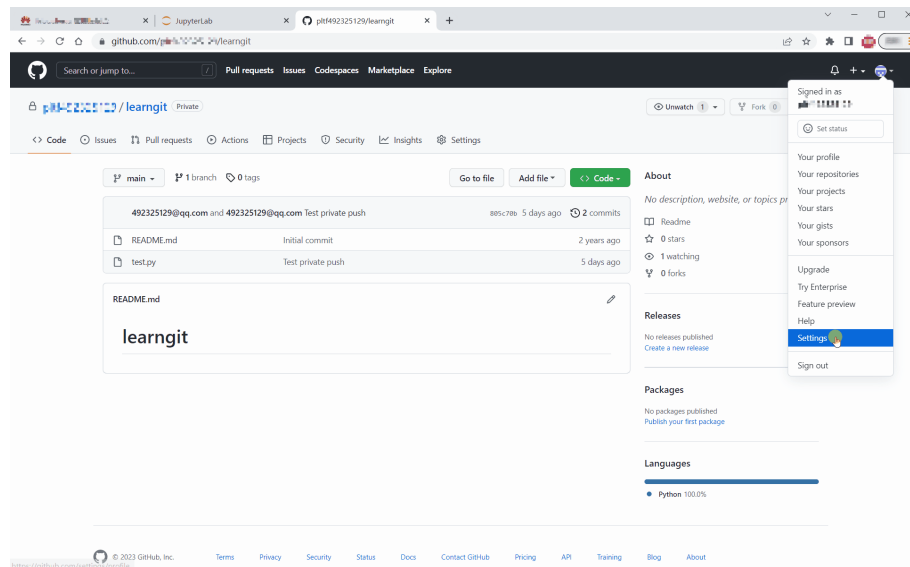


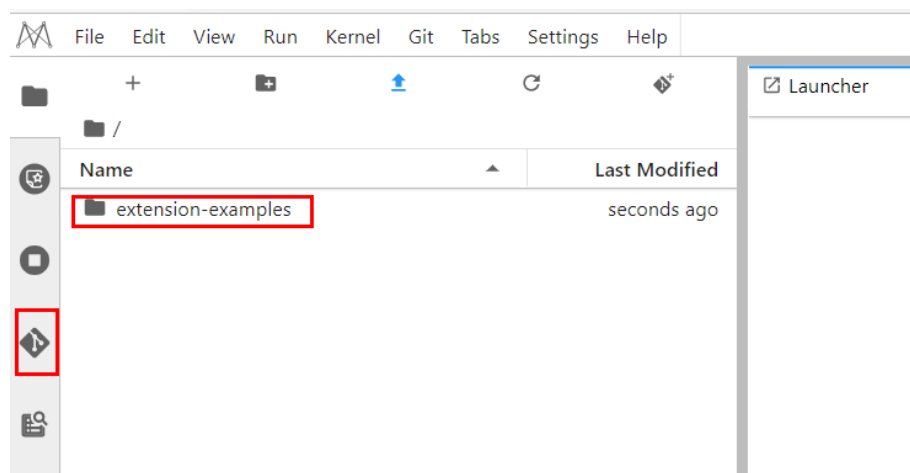
Figure 4-25 Obtaining a personal access token



Viewing a Code Repository

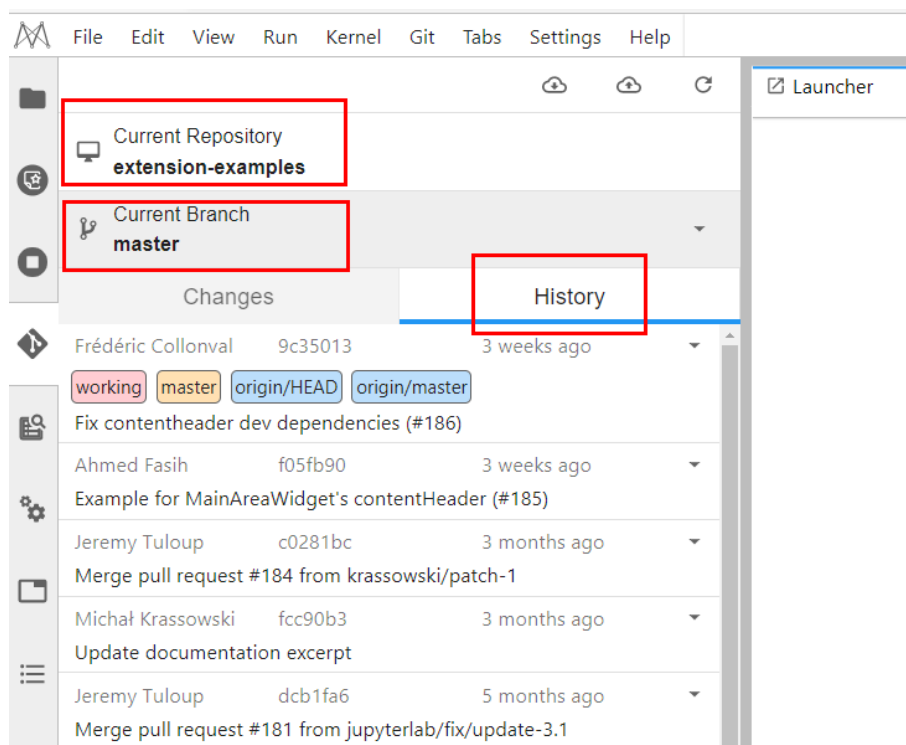
In the list under **Name**, double-click the folder you want to use and click the Git plug-in icon on the left to access the code repository corresponding to the folder.

Figure 4-26 Opening the folder and starting the Git plug-in



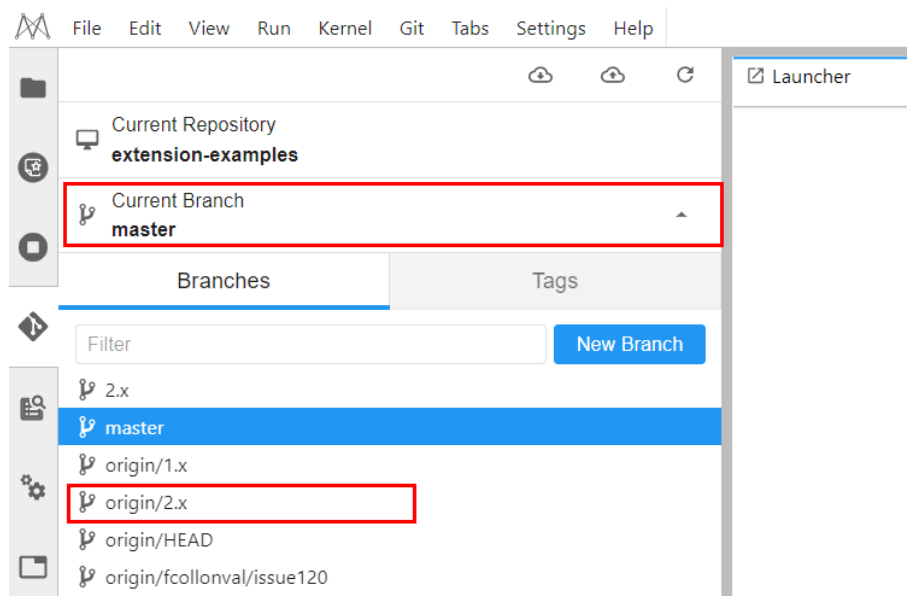
You can view the information current code repository, such as the repository name, branch, and historical submission records.

Figure 4-27 Viewing a code repository



NOTE

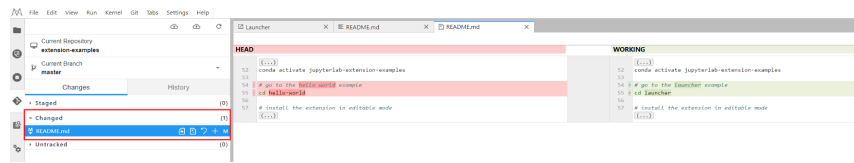
By default, the Git plug-in clones the master branch. To switch another branch, click **Current Branch** to expand all branches and click the target branch name.



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.

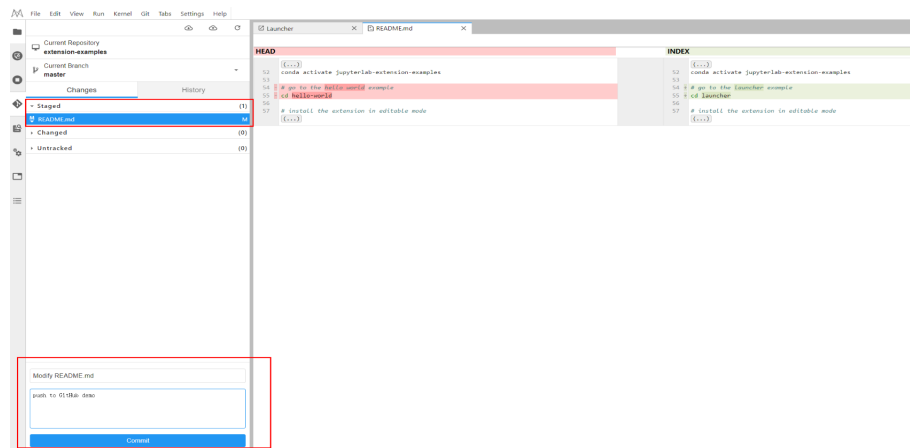
Figure 4-28 Viewing modifications



Committing Modifications

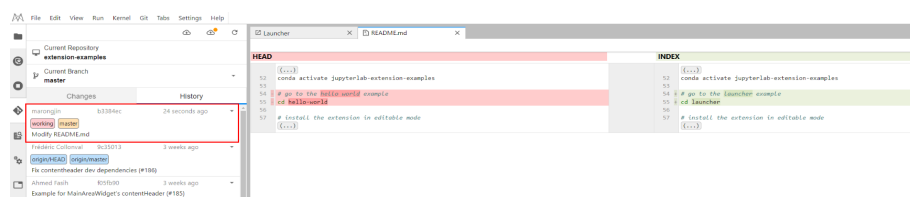
After confirming that the modifications are correct, click **Stage this change** on the right of the file name, which is equivalent to running the **git add** command. The file enters the **Staged** state. Enter the message to be committed in the lower left corner and click **Commit** that is equivalent to running the **git commit** command.

Figure 4-29 Committing modifications



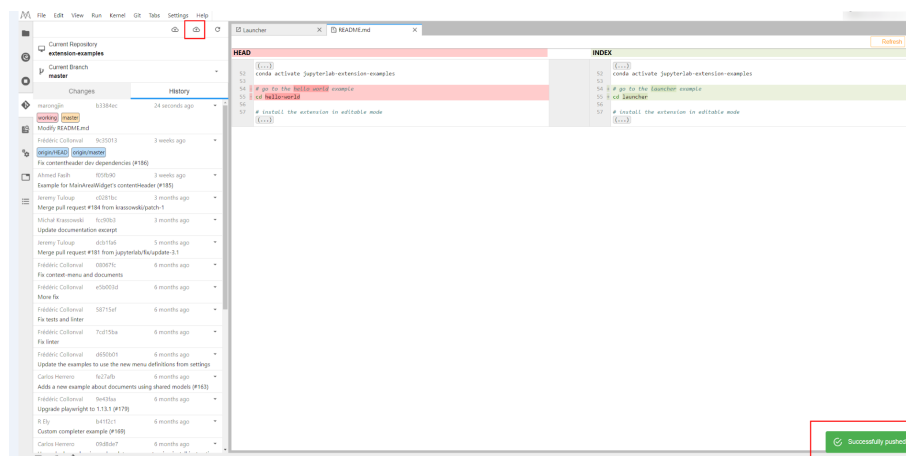
On the **History** tab page, view the committing status.

Figure 4-30 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 4-31 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.

4.6 Visualized Model Training

4.6.1 Introduction to Training Job Visualization

ModelArts notebook of the new version supports TensorBoard and MindInsight for visualizing training jobs. In the development environment, use small datasets to train and debug algorithms, during which you can check algorithm convergence and detect issues to facilitate debugging.

You can create visualization jobs of TensorBoard and MindInsight types on ModelArts.

Both TensorBoard and MindInsight effectively display the change trend of a training job and the data used in the training.

- TensorBoard**

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 2.1, and PyTorch 1.4 and 1.8 images. Select images and flavors based on the site requirements.
- MindInsight**

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](#).

The following shows the images and flavors supported by MindInsight visualization training jobs, and select images and flavors based on the site requirements.

- MindSpore 1.2.0 (CPU or GPU)

You can use the summary file generated during model training to create a visualization job in Notebook of DevEnviron.

- For details about how to create a MindInsight visualization job in a development environment, see [MindInsight Visualization Jobs](#).
- For details about how to create a TensorBoard visualization job in a development environment, see [TensorBoard Visualization Jobs](#).

4.6.2 MindInsight Visualization Jobs

ModelArts notebook of the new version 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, choose **DevEnviron > Notebook**, and create a development environment instance for the MindSpore engine. After the instance is created, click **Open** in the **Operation** column of the instance to access it online.

The images and resource types supported by MindInsight visualization training jobs are as follows:

- MindSpore 1.2.0 (CPU or GPU)
- MindSpore 1.5.x or later (Ascend)

Step 2 Upload the Summary Data

Summary data is required for MindInsight visualization in a development environment.

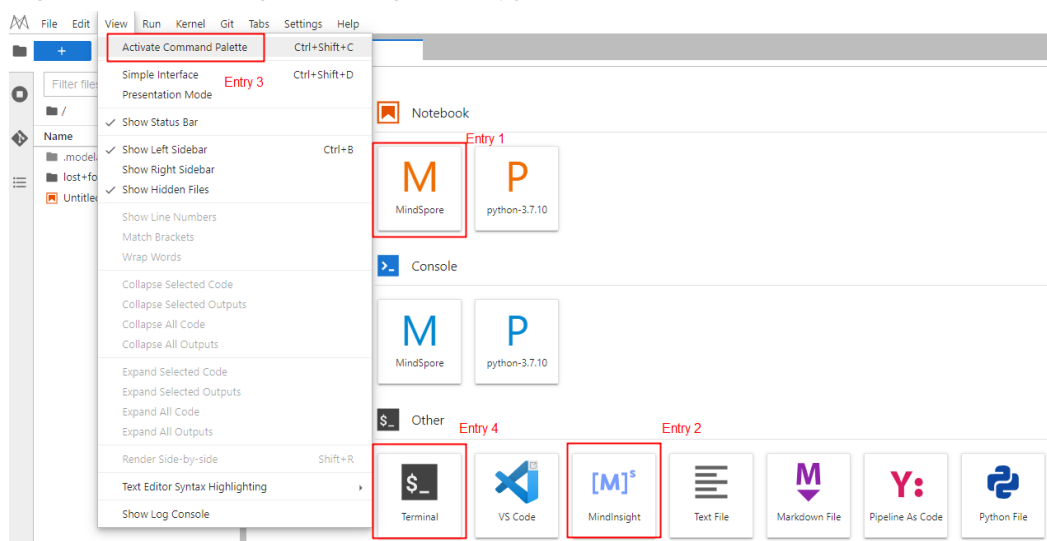
Upload the summary data to the **/home/ma-user/work/** directory in a development environment or store it in an OBS parallel file system.

- For details about how to upload the summary data to **/home/ma-user/work/**, see [Uploading Files 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

Choose a way you like to start MindInsight in JupyterLab.

Figure 4-32 Starting MindInsight in JupyterLab



Method 1



1. Click [MindSpore](#) to go to the JupyterLab development environment. An IPYNB file will be automatically created.
2. Enter the following command in the dialog box:

```
%reload_ext mindinsight  
%mindinsight --port {PORT} --summary-base-dir {SUMMARY_BASE_DIR}
```

Parameters:

- **port** *{PORT}*: web service port for visualization, which defaults to **8080**. If the default port **8080** has been used, specify a port ranging from 1 to 65535.
- **summary-base-dir** *{SUMMARY_BASE_DIR}*: data storage path in the development environment
 - Local path to the development environment: **./work/xxx** (relative path) or **/home/ma-user/work/xxx** (absolute path)
 - Path to the OBS parallel file system bucket: **obs://xxx/**

For example:

```
# If the summary data is stored in /home/ma-user/work/ of a development environment, run the following command:
```

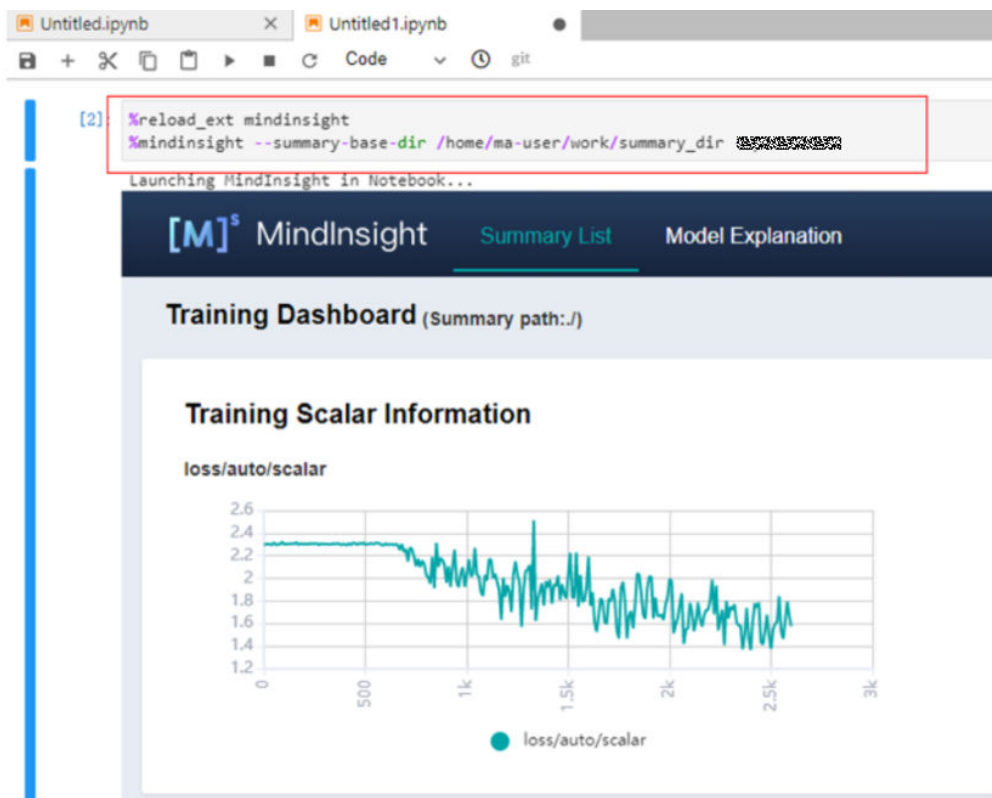
```
%mindinsight --summary-base-dir /home/ma-user/work/xxx
```

Or

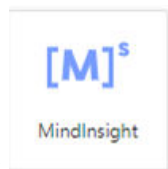
```
# If the summary data is stored in an OBS parallel file system, run the following command. Then, the development environment will automatically mount the storage path to the OBS parallel file system and read data from the path.
```

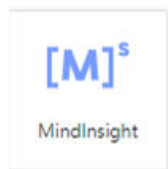
```
%mindinsight --summary-base-dir obs://xxx/
```

Figure 4-33 MindInsight page (1)



Method 2

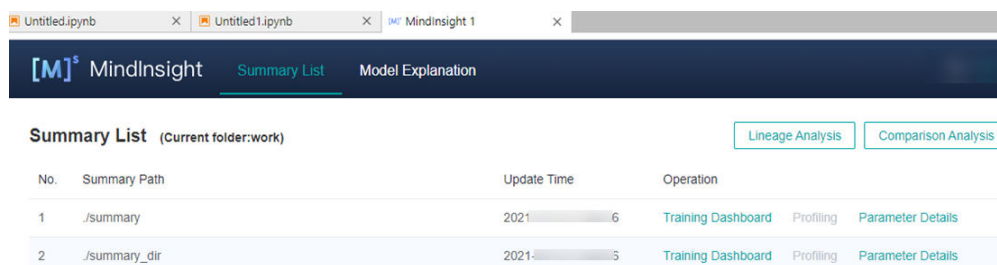


Click  to go to the MindInsight page.

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.

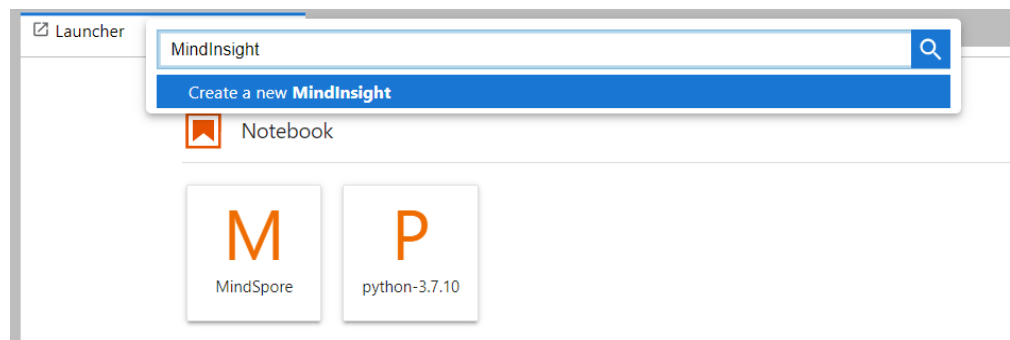
Figure 4-34 MindInsight page (2)



Method 3

1. Choose **View > Activate Command Palette**, enter **MindInsight** in the search box, and click **Create a new MindInsight**.

Figure 4-35 Create a new MindInsight



2. Enter the path to the summary data or the storage path to the OBS parallel file system, and click **CREATE**.
 - Local path to the development environment: **./summary** (relative path) or **/home/ma-user/work/summary** (absolute path)
 - Path to the OBS parallel file system: **obs://xxx/**

Figure 4-36 Path to the summary data

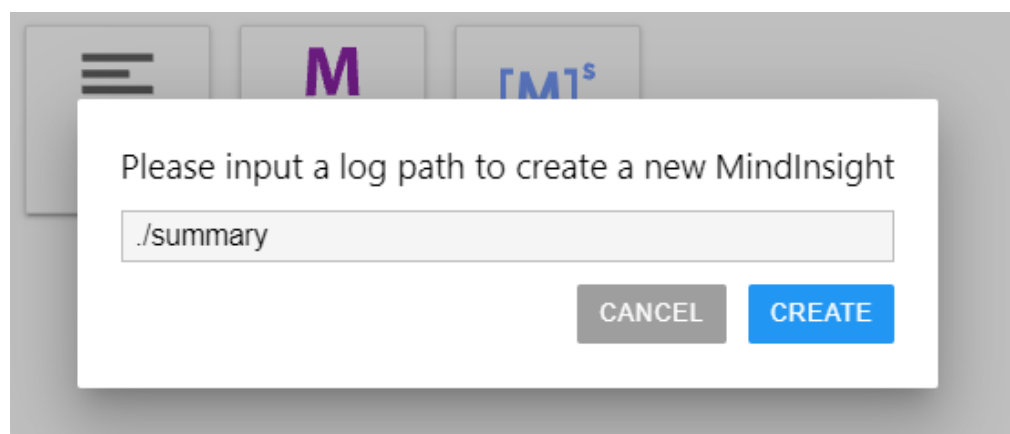
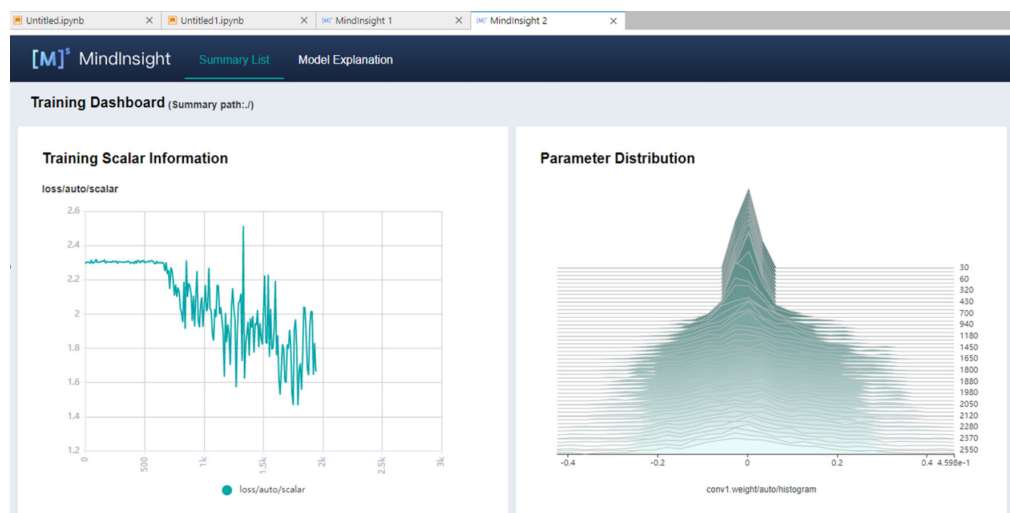


Figure 4-37 MindInsight page (3)



 **NOTE**

A maximum of 10 MindInsight instances can be started using method 2 or 3.

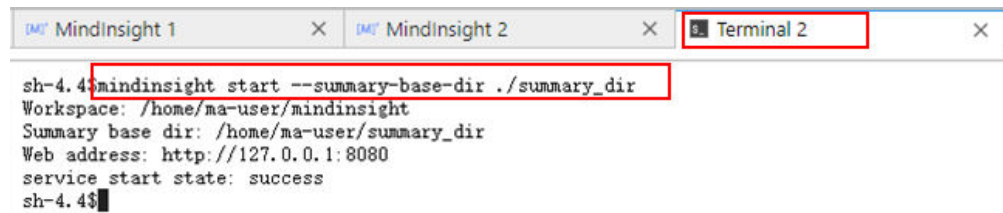
Method 4



Click **Terminal** and run the following command (the UI will not be displayed):

```
mindinsight start --summary-base-dir ./summary_dir
```

Figure 4-38 Opening MindInsight through Terminal



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.

For more information, see [Viewing Training Dashboard](#) on the MindSpore official website.

Related Operations

To stop a MindInsight instance, use one of the following methods:

- Method 1: Enter the following command in the **.ipynb** file window of JupyterLab. in which the port number is configured in **Start MindInsight (8080 by default)**:


```
!mindinsight stop --port 8080
```
- Method 2: 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 4-39 Stopping an instance




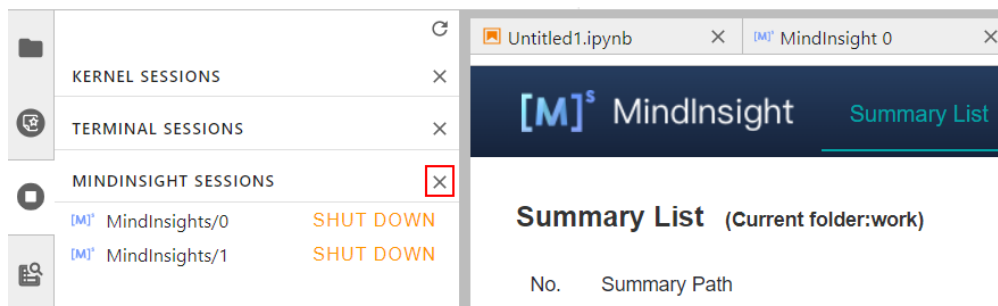
- Method 3: Click  in the following figure to close all started MindInsight instances.

Figure 4-40 Stopping all started MindInsight instances



- Method 4 (not recommended): Close the MindInsight window on JupyterLab. In this way, only the visualization window is closed, but the instance is still running on the backend.

4.6.3 TensorBoard Visualization Jobs

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.

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](#).

Note

- 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

On the ModelArts management console, choose **DevEnviron** > **Notebook**, and 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.

Only CPU and GPU flavors with TensorFlow2.1, PyTorch1.4, or PyTorch1.8 and later images can support TensorBoard visualization for training jobs. 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.

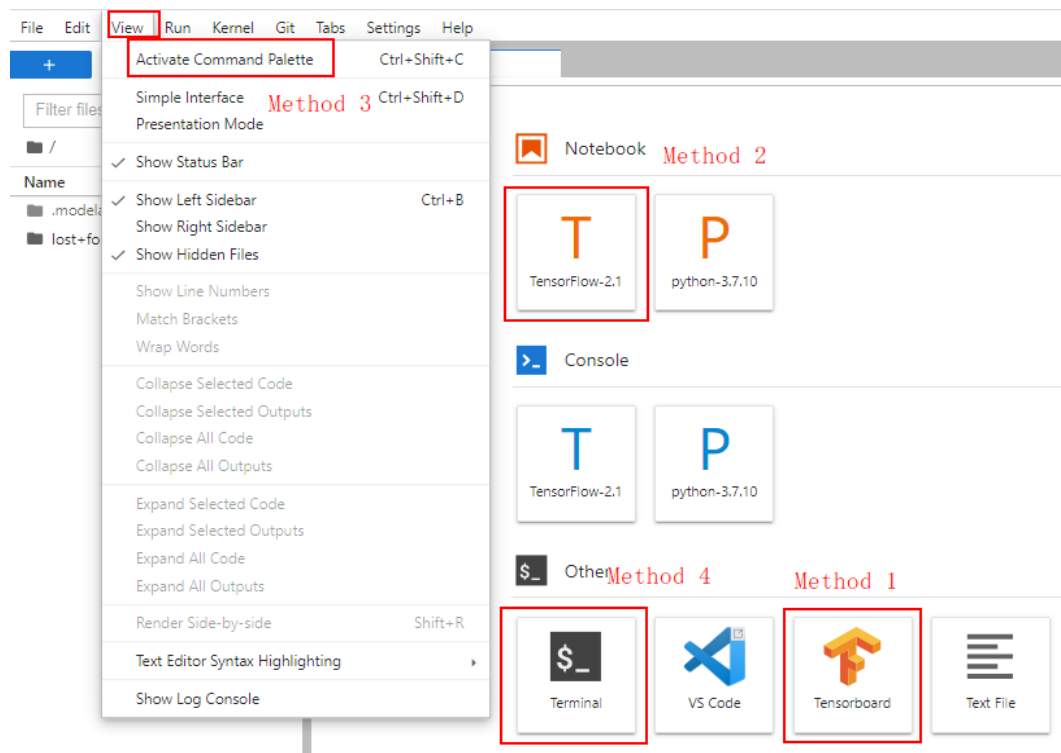
You can upload the summary data to the `/home/ma-user/work/` directory in the development environment or store it in the OBS parallel file system.

- For details about how to upload the summary data to the notebook path `/home/ma-user/work/`, see [Uploading Files 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

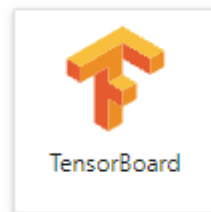
There are multiple methods to open TensorBoard in JupyterLab in the development environment. Select one based on your habits.

Figure 4-41 Starting TensorBoard in JupyterLab



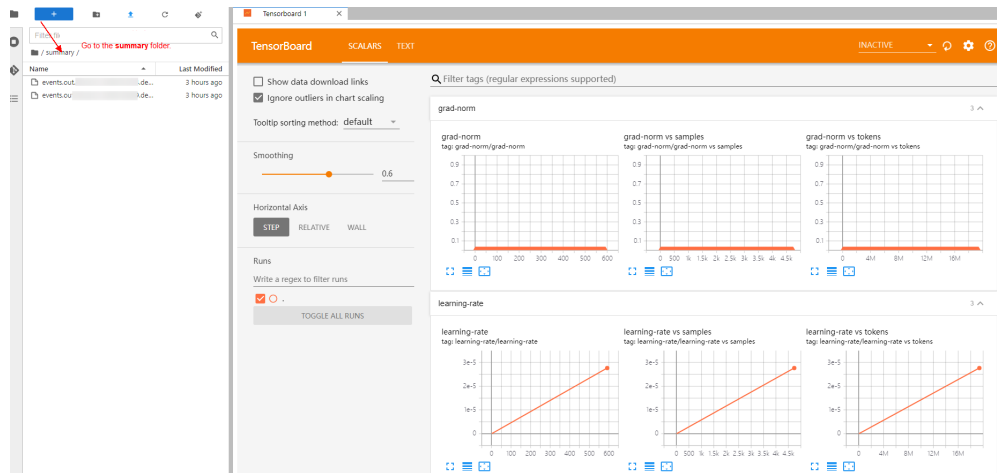
Method 1 (recommended):

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 4-42](#). to go to the

Figure 4-42 TensorBoard page (1)




Method 2

NOTICE

You can upgrade TensorBoard to any version except 2.4.0. After the upgrade, the new version of TensorBoard is used only in method 2. For other methods, use TensorBoard 2.1.1.



1. Click  to go to the JupyterLab development environment. The .ipynb file is automatically created.
2. Enter the following command in the dialog box:

```
%reload_ext ma_tensorboard
%ma_tensorboard --port {PORT} --logdir {BASE_DIR}
```

Parameters:

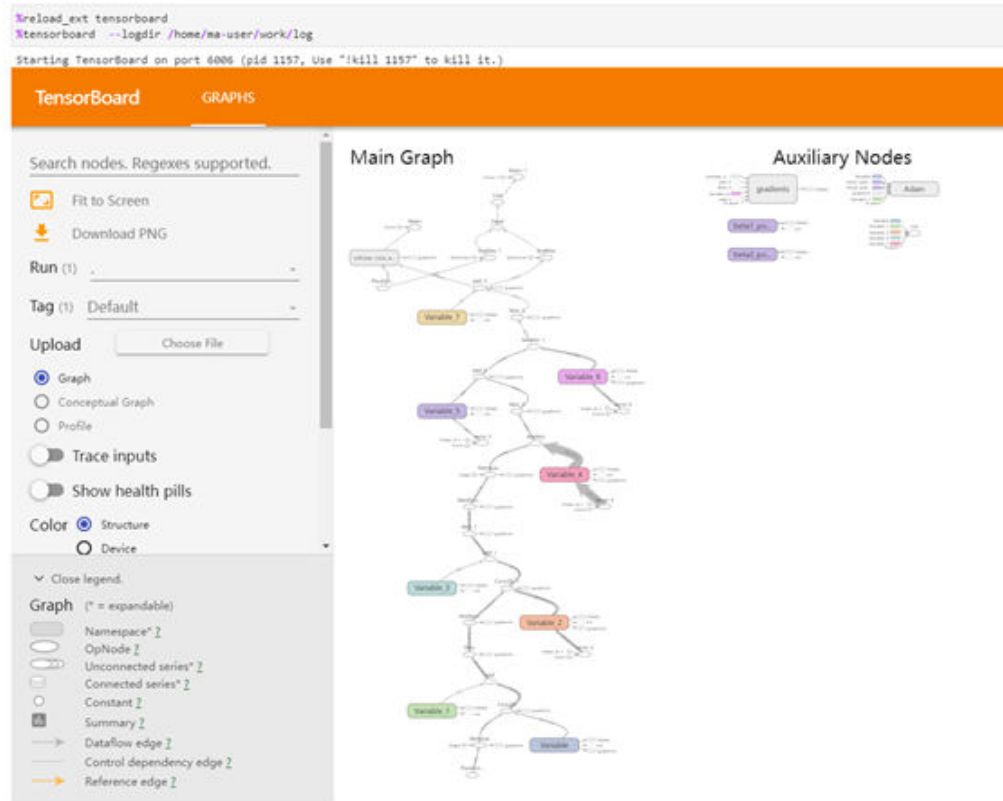
- **port {PORT}**: web service port for visualization, which defaults to **8080**. If the default port **8080** has been used, specify a port ranging from 1 to 65535.
- **logdir {BASE_DIR}**: data storage path in the development environment
 - Local path of the development environment: **./work/xxx** (relative path) or **/home/ma-user/work/xxx** (absolute path)
 - Path of the OBS parallel file system: **obs://xxx/**

Example:

```
# If the summary data is stored in /home/ma-user/work/ of the development environment, run the
following command:
%ma_tensorboard --port {PORT} --logdir /home/ma-user/work/xxx
or
# If the summary data is stored in the OBS parallel file system, run the following command and the
development environment automatically mounts the storage path of the OBS parallel file system and
```

```
reads data.
%ma_tensorboard --port {PORT} --logdir obs://xxx/
```

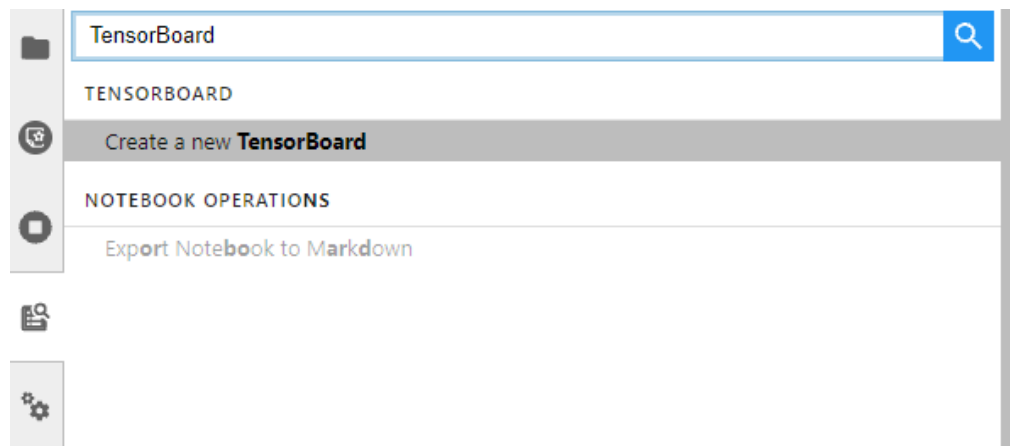
Figure 4-43 TensorBoard page (2)



Method 3

1. Choose **View > Activate Command Palette**, enter **TensorBoard** in the search box, and click **Create a new TensorBoard**.

Figure 4-44 Create a new TensorBoard



2. Enter the path of the summary data you want to view or the storage path of the OBS parallel file system.

- Local path of the development environment: **./summary** (relative path) or **/home/ma-user/work/summary** (absolute path)
- Path of the OBS parallel file system bucket: **obs://xxx/**

Figure 4-45 Entering the summary data path

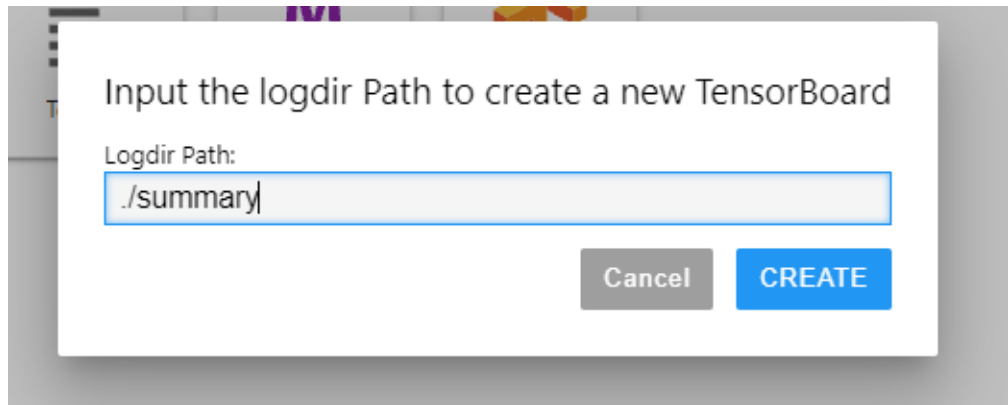
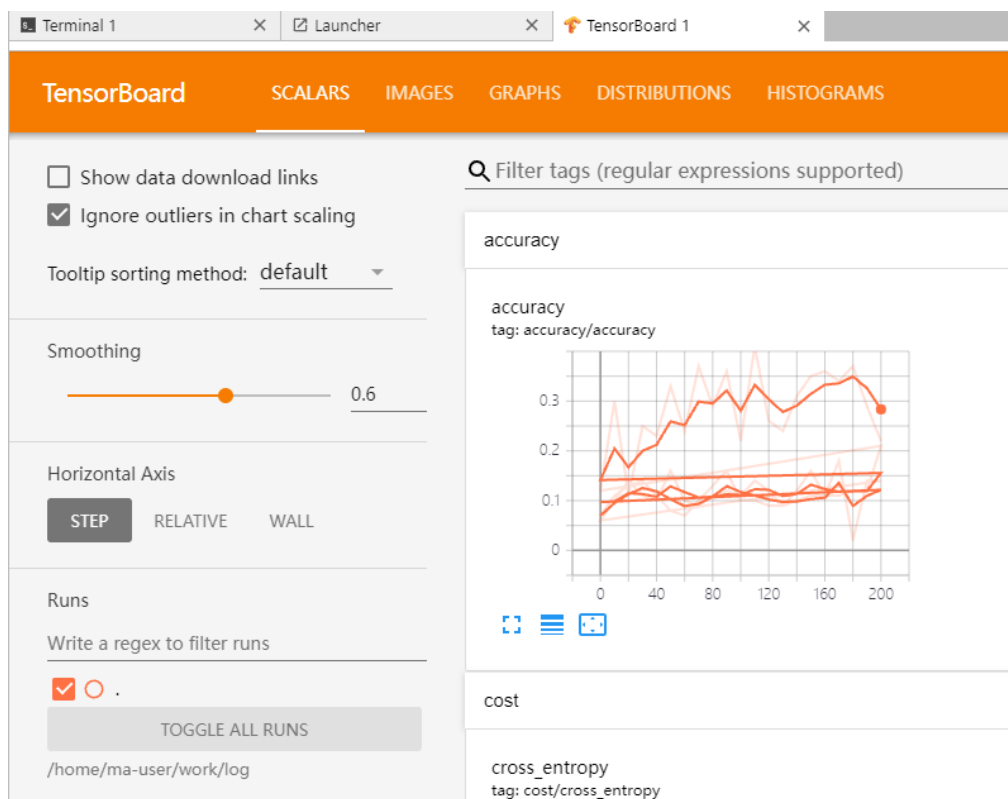


Figure 4-46 TensorBoard page (3)



Method 4



Click **Terminal** and run the following command. The UI will not be displayed.


```
tensorboard --logdir ./log
```

Figure 4-47 Opening TensorBoard through Terminal

```
sh-4.4$pwd
/home/aa/user
sh-4.4$tensorboard --logdir ./log
2021-10-18 20:34:53.584976: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libxvner.so.6
2021-10-18 20:34:53.589272: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libxvner_plugin.so.6
Serving TensorBoard on localhost; to expose to the network, use a proxy or pass --bind_all
TensorBoard 2.1.1 at http://localhost:6006/ (Press CTRL+C to quit)
```

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](#).

Related Operations

To stop a TensorBoard instance, use any of the following methods:


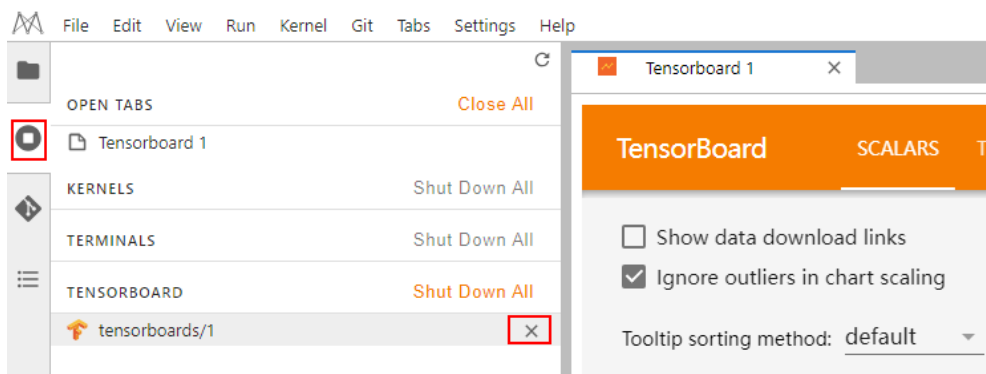
- Method 1: Click . The TensorBoard instance management page is displayed, which shows all started TensorBoard instances. Click **SHUT DOWN** next to an instance.

Figure 4-48 Clicking SHUT DOWN to stop an instance



- Method 2: Enter the following command in the .ipynb file window in JupyterLab (Obtain PID on the startup screen or using the command **ps -ef | grep tensorboard**):
!kill PID


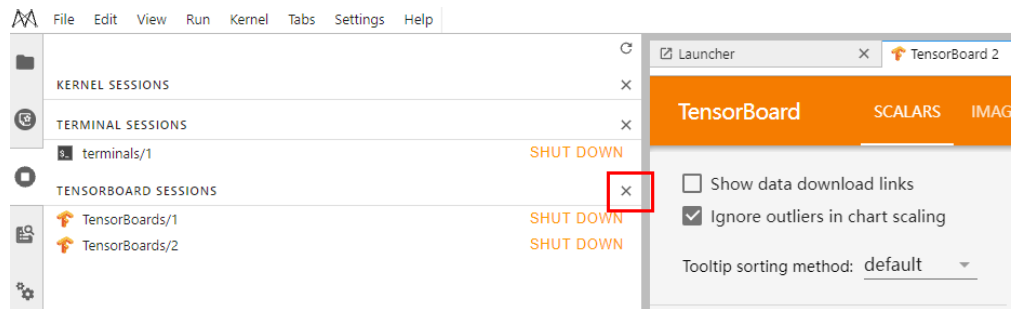
- Method 3: Click  as shown in the following figure to stop all started TensorBoard instances.

Figure 4-49 Stopping all started TensorBoard instances



- (Not recommended) Method 4: Close the TensorBoard window in JupyterLab. This method closes only the window, but the instance is still running on the backend.

4.7 Uploading and Downloading Data in Notebook

4.7.1 Uploading Files to JupyterLab

4.7.1.1 Scenarios

Easy and fast file uploading is a common requirement in AI development.

Before the optimization, ModelArts only allowed local files not exceeding 100 MB to be directly uploaded to a notebook instance. However, the files to be uploaded are not all stored locally, which may be from an open-source repository of GitHub, an open-source dataset (<https://nodejs.org/dist/v12.4.0/node-v12.4.0-linux-x64.tar.xz>), or OBS. Additionally, ModelArts did not show the file uploading progress or speed.

ModelArts has been optimized for better file uploading experience. It not only provides more file upload functions, but also displays more file upload details.

Optimized file uploading:

- Supports local files.
- Supports cloning files from open-source repositories in GitHub.
- Supports OBS files.
- Supports remote files.
- Supports visualized upload progress.

4.7.1.2 Uploading Files from a Local Path to JupyterLab

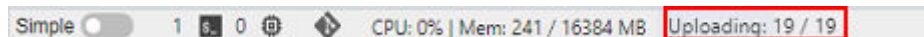
4.7.1.2.1 Upload Scenarios and Entries

JupyterLab provides multiple methods for uploading files.

Methods for Uploading a File

- For a file that does not exceed 100 MB, directly upload it, and details such as the file size, upload progress, and upload speed are displayed.

- For a file that exceeds 100 MB but does not exceed 5 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 5 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.



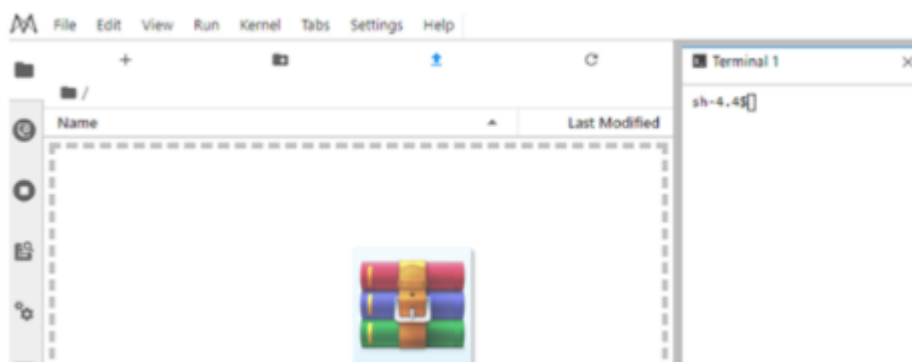
Simple 1 0 CPU: 0% | Mem: 241 / 16384 MB Uploading: 19 / 19

Prerequisites

You have used JupyterLab to open a running notebook environment.

Upload Entry 1: Dragging a File to the File Browser Window

Drag the file to the blank area on the left of the JupyterLab window and upload it.



Upload Entry 2: Clicking the File Upload Icon and Uploading a File


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 4-50 File upload icon

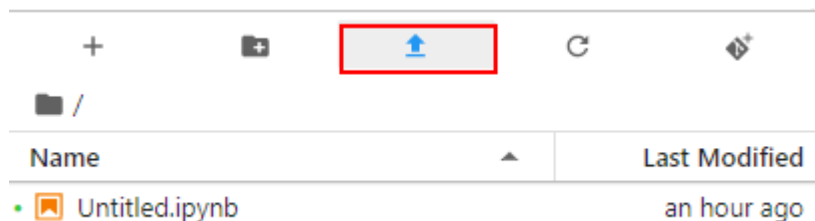
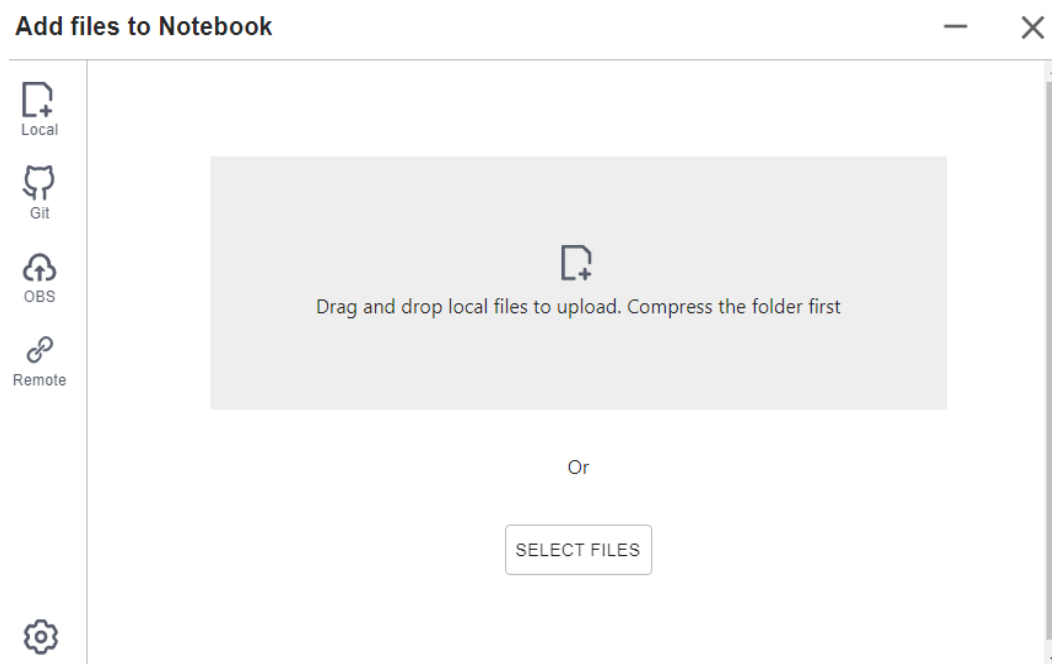


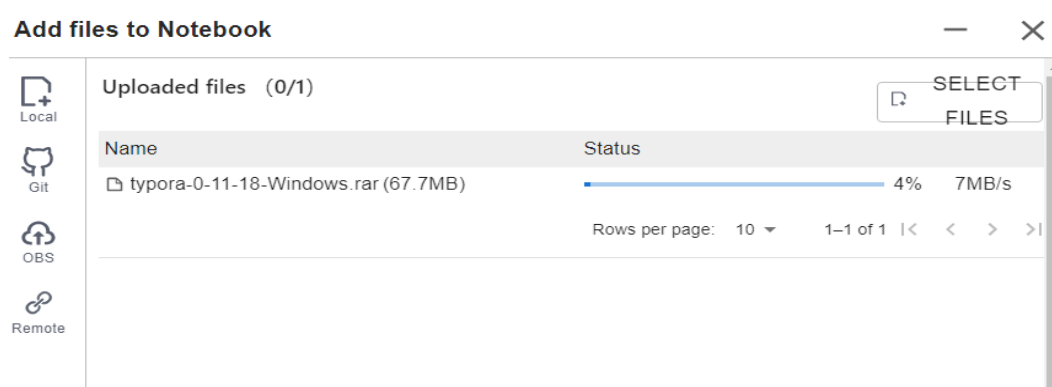
Figure 4-51 File upload page



4.7.1.2.2 Uploading a Local File Less Than 100 MB to JupyterLab

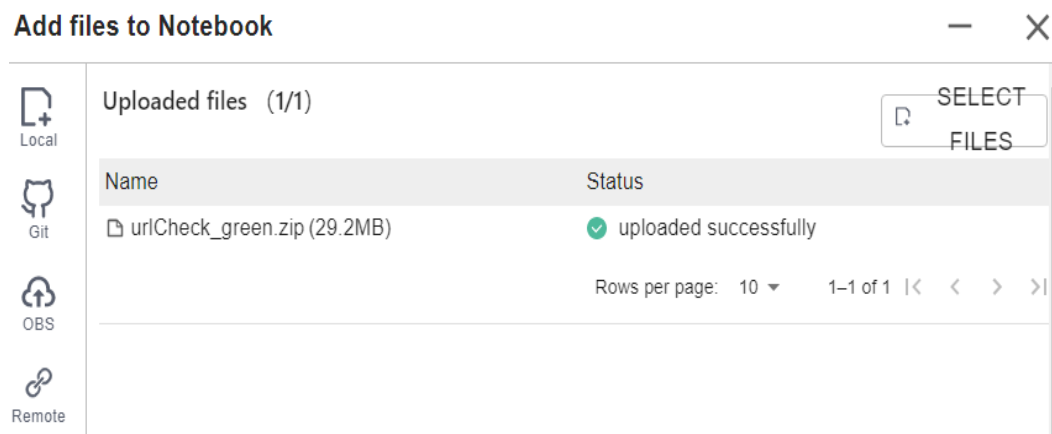
For a file not exceeding 100 MB, directly upload it to the target notebook instance. Detailed information, such as the file size, upload progress, and upload speed are displayed.

Figure 4-52 Uploading a file less than 100 MB



A message is displayed after the file is uploaded.

Figure 4-53 Uploaded

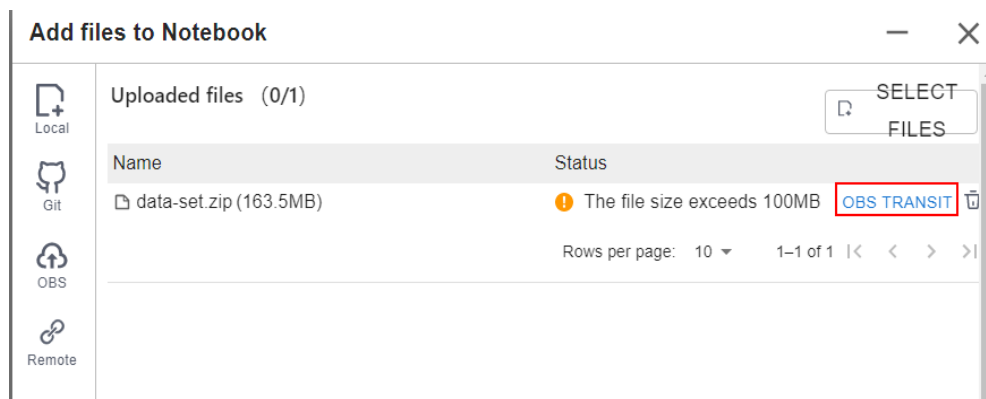


4.7.1.2.3 Uploading a Local File with a Size Ranging from 100 MB to 5 GB to JupyterLab

For a file that exceeds 100 MB but does not exceed 5 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.

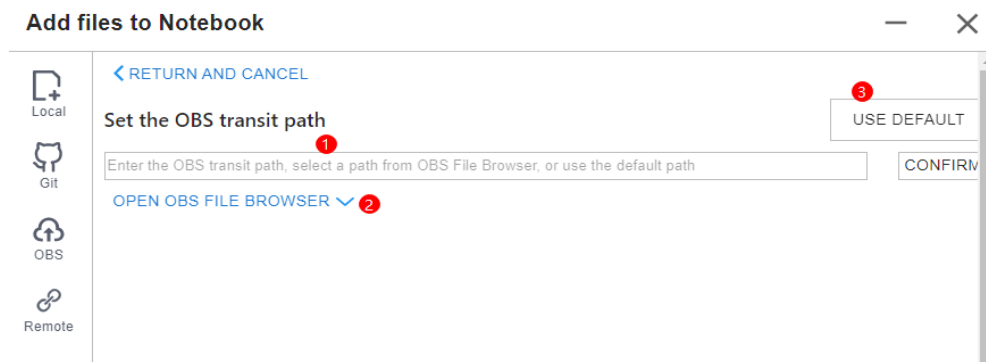
For example, in the scenario shown in the following figure, upload the file through OBS.

Figure 4-54 Uploading a large file through OBS




To upload a large file through OBS, set an OBS path.

Figure 4-55 Uploading a file through OBS

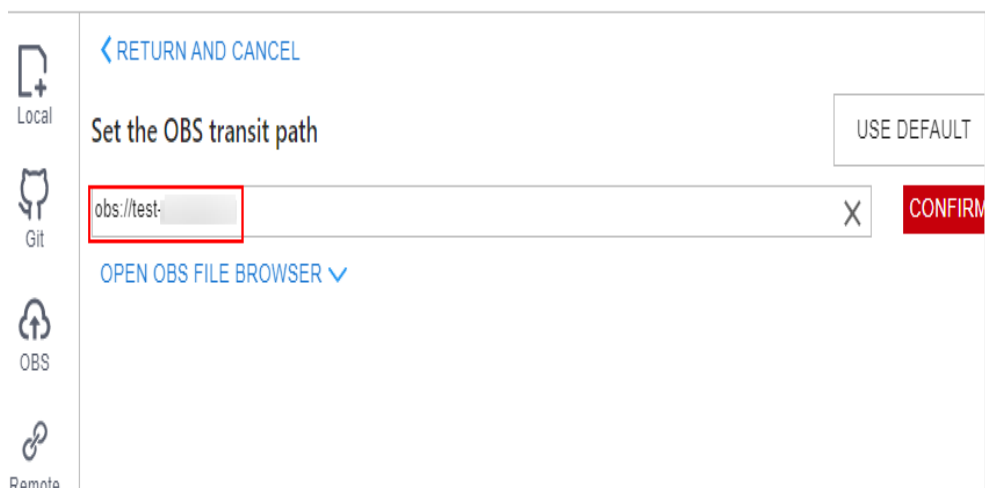


NOTE

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.

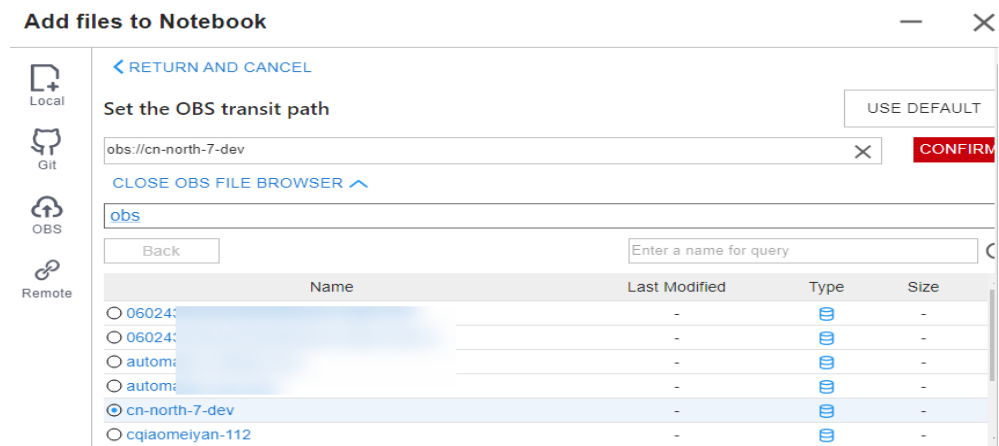
- Method 1: Enter a valid OBS path in the text box and click **OK**.

Figure 4-56 Configuring an OBS path



- Method 2: Select an OBS path in **OBS File Browser** and click **OK**.

Figure 4-57 OBS File Browser



- Method 3: Use the default path.

Figure 4-58 Using the default path to upload a file

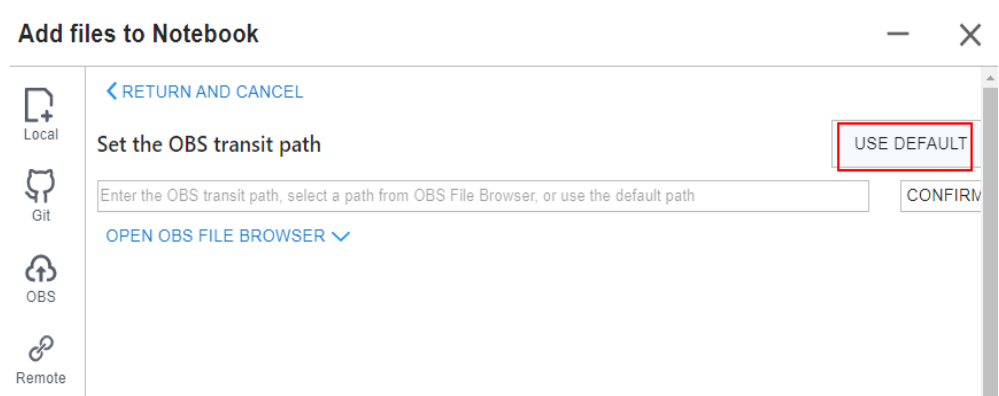
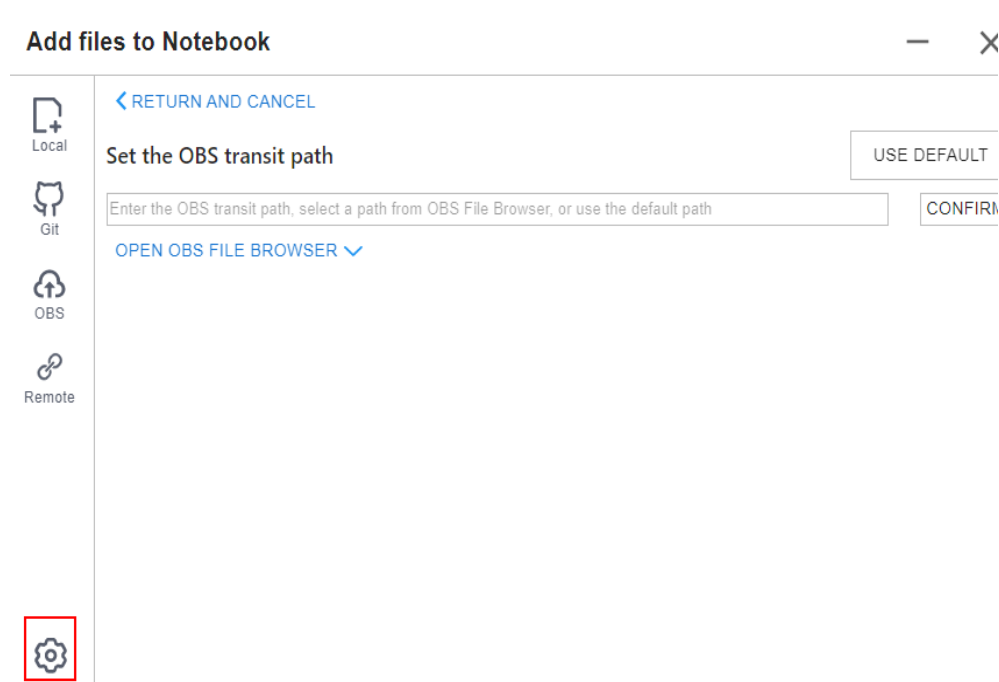
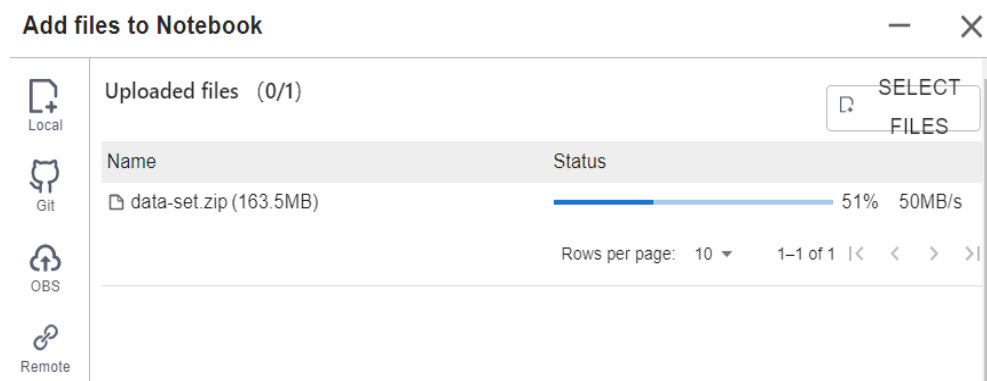


Figure 4-59 Setting an OBS path for uploading a local file



After the OBS path is set, upload a file.

Figure 4-60 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.
```

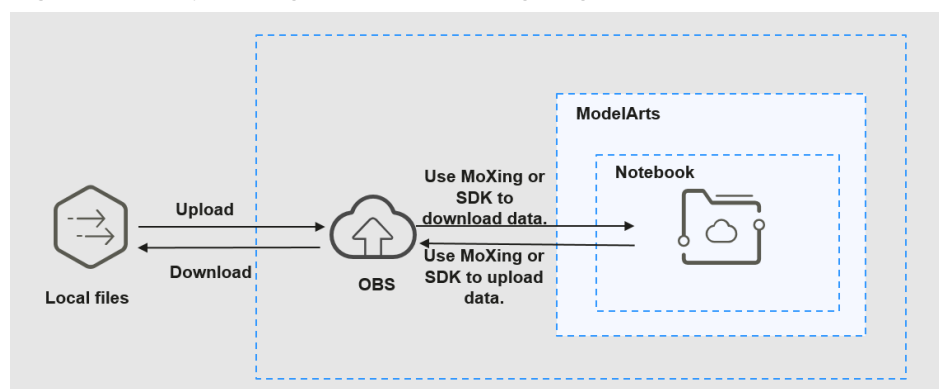
For more details, search for the decompression command in mainstream search engines.

4.7.1.2.4 Uploading a Local File Larger Than 5 GB to JupyterLab

A file exceeding 5 GB cannot be directly uploaded to JupyterLab.

To upload files exceeding 5 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 4-61 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 4-62](#) and run the **ls /home/ma-user/work** command to view the file downloaded to the notebook instance. Alternatively, view the downloaded file in the left navigation pane of Jupyter. If the file is not displayed, refresh the page.

Figure 4-62 Opening the terminal

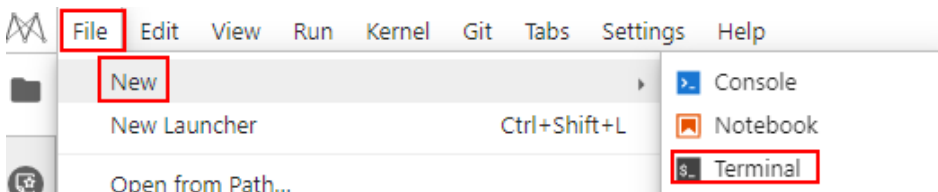
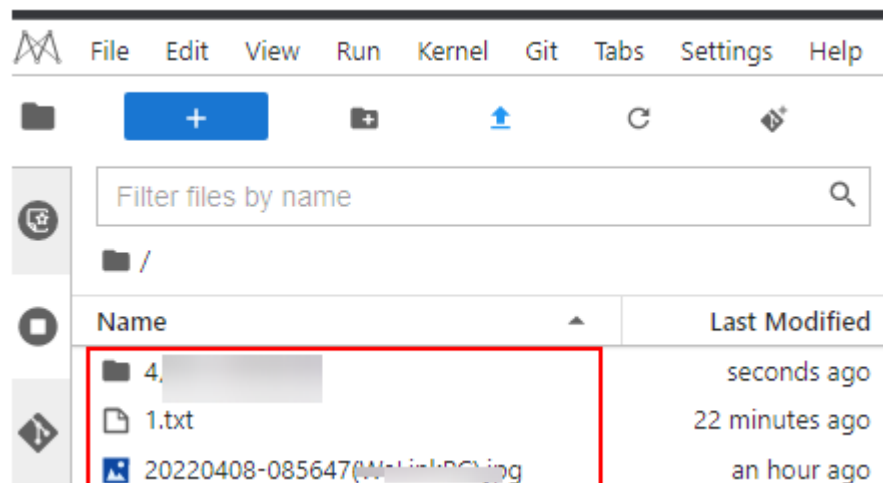


Figure 4-63 File downloaded to a notebook instance



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 the OBS bucket and notebook instance are in different regions, the access to OBS is denied.
- Ensure that the notebook account has the permission to read data in the OBS bucket.

For details, see [Incorrect OBS Path on ModelArts](#).

4.7.1.3 Cloning an Open-Source Repository in GitHub

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 4-64 File upload icon

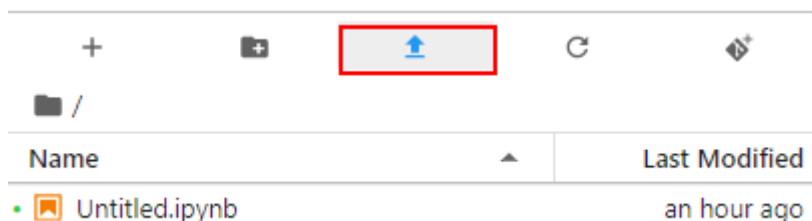
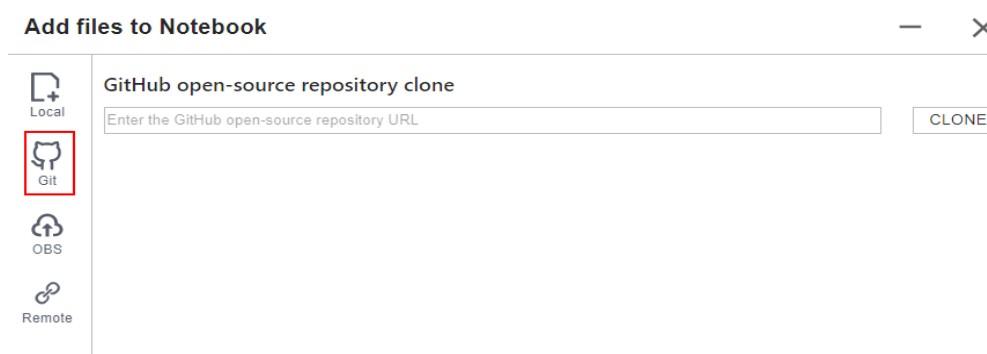


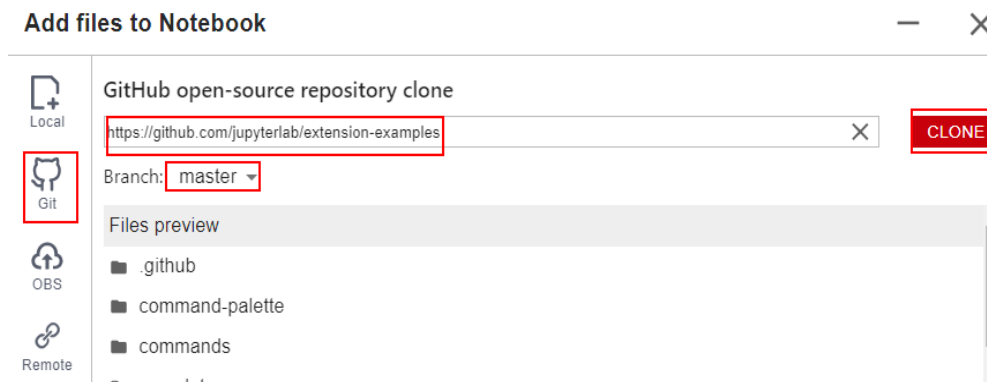
Figure 4-65 Page for cloning files from a GitHub open-source repository



3. Enter a valid address of a GitHub open-source repository, select files from the displayed files and folders, and click **Clone**.

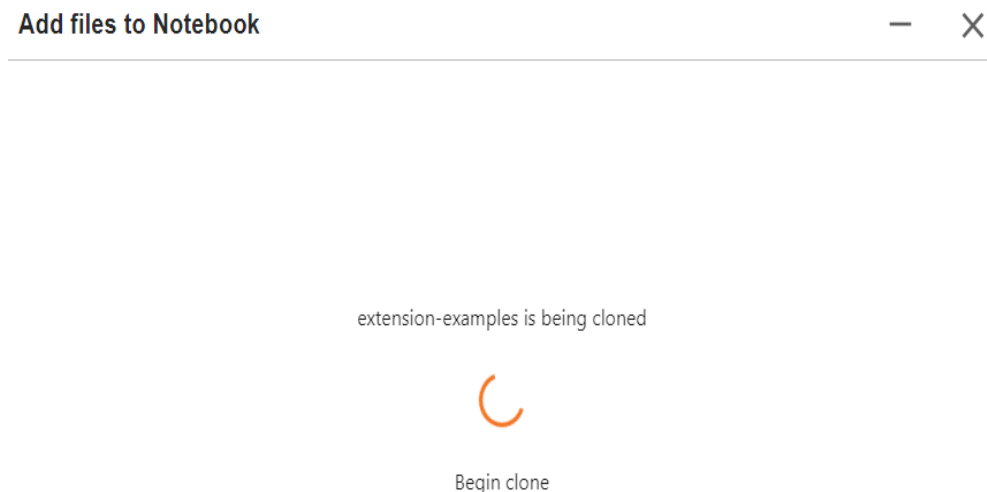
GitHub open-source repository address: <https://github.com/jupyterlab/extension-examples>

Figure 4-66 Entering a valid address of a GitHub open-source repository




4. View the clone process.

Figure 4-67 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.

4.7.1.4 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.



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 OBS file upload page.

Figure 4-68 File upload icon

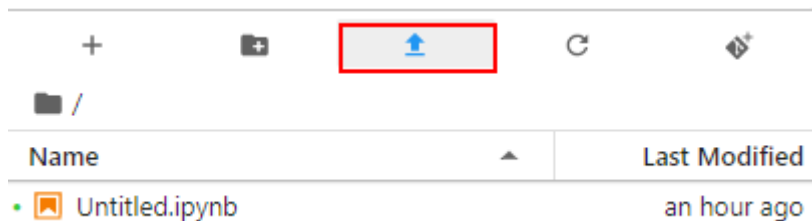
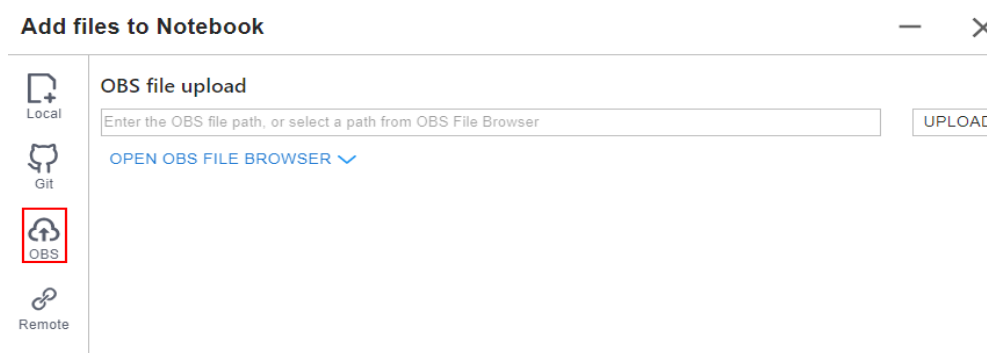
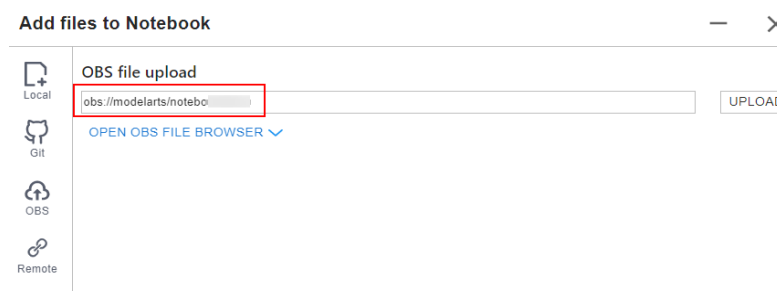


Figure 4-69 OBS file upload



3. 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 4-70 Entering a valid OBS file path

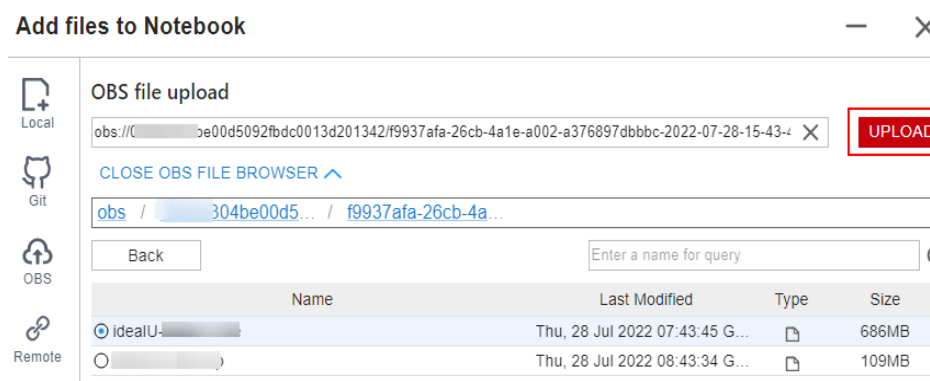


 **NOTE**

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 4-71 Uploading an OBS File



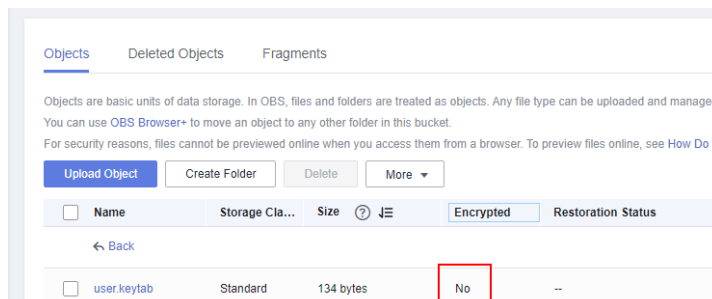
Error Handling

There are three typical scenarios in which uploading a file failed.

- **Scenario 1**

Possible causes:

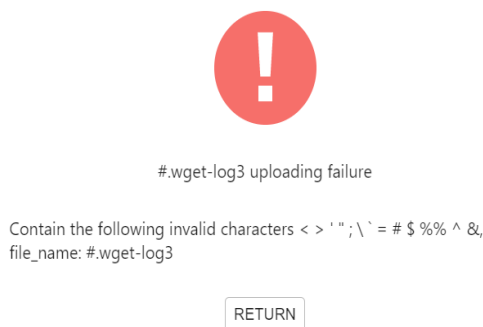
- The OBS path is set to a folder instead of a file path.
- The file in OBS is encrypted. In this case, go to the OBS console and ensure that the file is encrypted.



- 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 4-72 File uploading failure

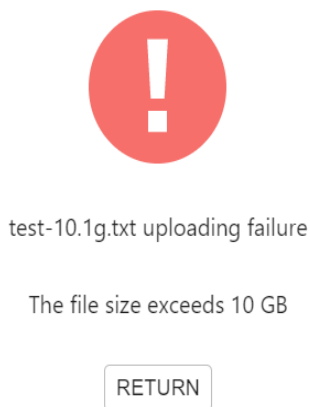


Possible causes:

The file name contains special characters such as <>'";\`=#\$%^&.

- **Scenario 3**

Figure 4-73 File uploading failure



Possible causes:

The uploaded file exceeded 10 GB.

4.7.1.5 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 4-74 File upload icon

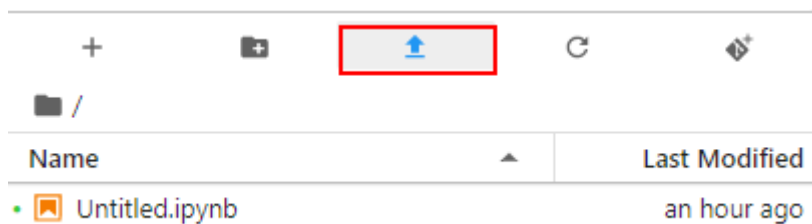
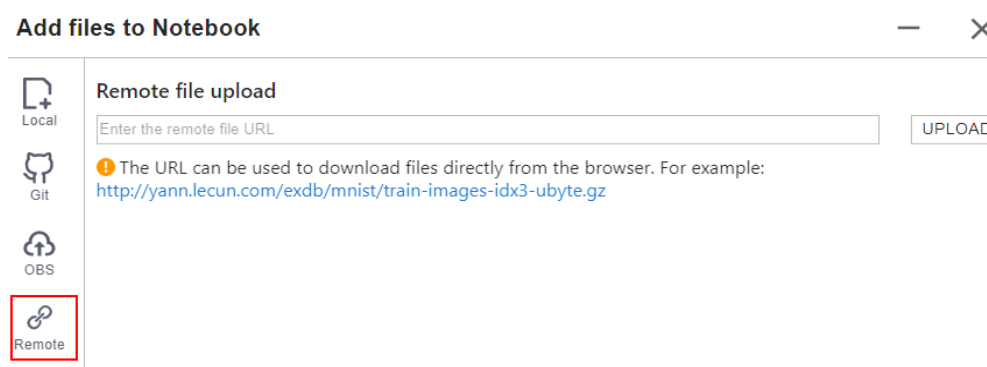
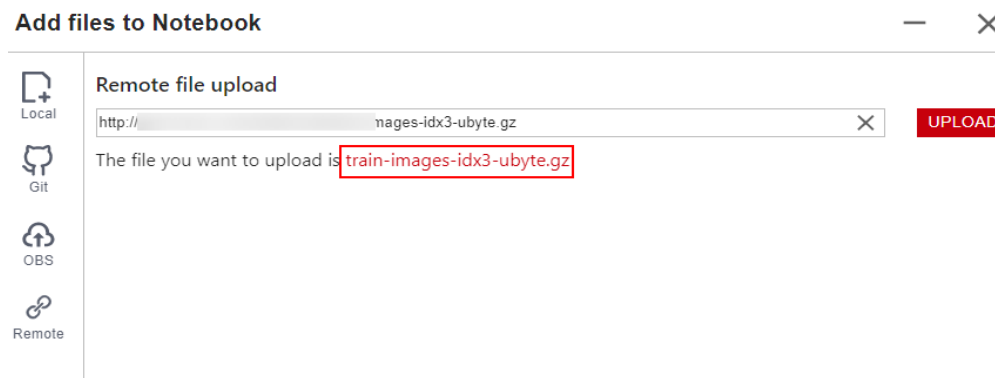


Figure 4-75 Remote file upload page



3. Enter a valid remote file URL, and the system automatically identifies the file name. Then, click **Upload**.

Figure 4-76 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.

4.7.2 Downloading a File from JupyterLab to a Local Path

Files created in JupyterLab can be downloaded to a local path.

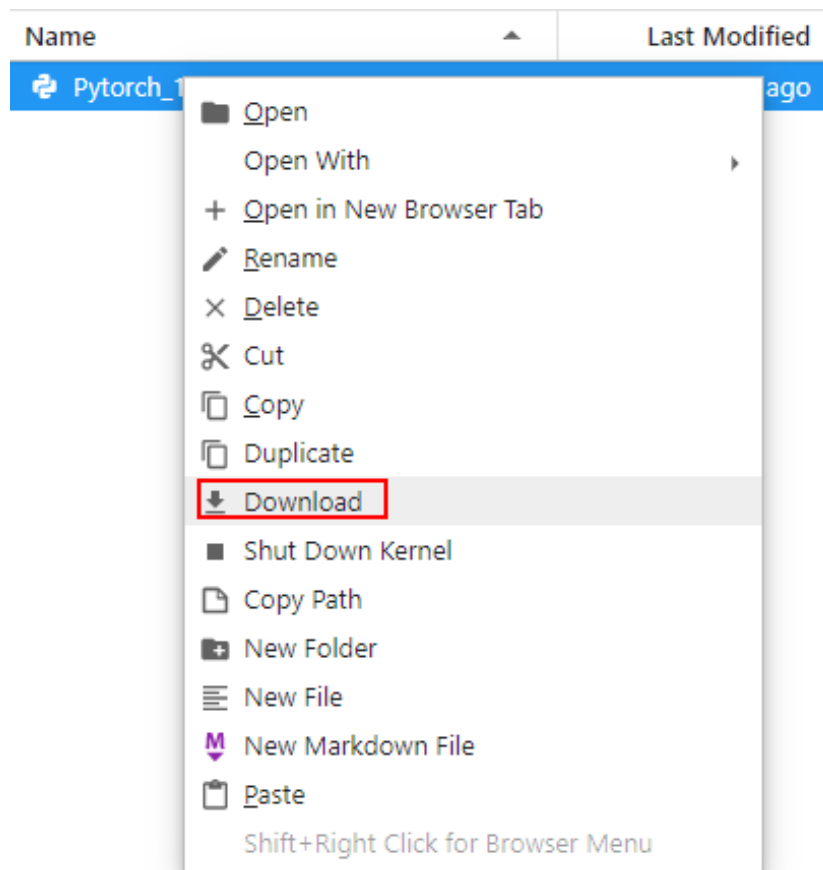
- If a file is less than or equal to 100 MB, directly download it from JupyterLab. For details, see [Downloading a File Less Than or Equal to 100 MB](#).

- 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 4-77 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. In the notebook instance, create an IPYNB file larger than 100 MB and use MoXing to upload it to OBS. Example code is as follows:

```
import moxing as mox
mox.file.copy('/home/ma-user/work/obs_file.txt', 'obs://bucket_name/obs_file.txt')
```

/home/ma-user/work/obs_file.txt is the path to the file stored in the notebook instance. **obs://bucket_name/obs_file.txt** is the path of the file uploaded to OBS, where **bucket_name** is the name of the bucket created in OBS, and **obs_file.txt** is the uploaded file.

2. 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/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 Local IDE

Operation Process in a Local IDE

[Local IDE \(PyCharm\)](#)

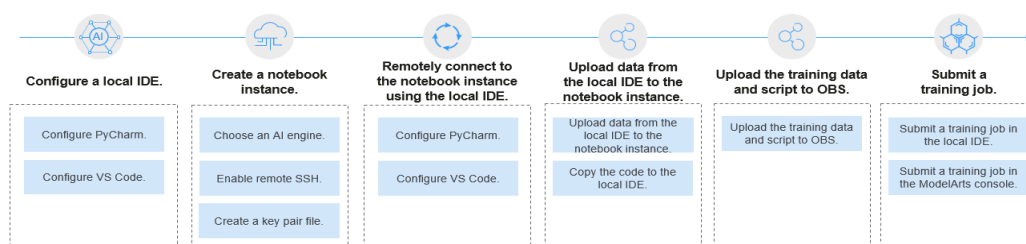
[Local IDE \(VS Code\)](#)

[Local IDE \(Accessed Using SSH\)](#)

5.1 Operation Process in a Local IDE

ModelArts allows you to remotely access notebook instances from a local IDE to develop AI models based on PyTorch, TensorFlow, or MindSpore. The following figure shows the operation process.

Figure 5-1 Using a local IDE to develop models



1. Configure a local IDE.

Configure a local IDE on your PC.

You can use **PyCharm**, **VS Code**, or **SSH tools** to access a notebook instance from a local IDE. PyCharm and VS Code can be automatically configured using plug-ins or manually configured.

2. Create a notebook instance.

On the ModelArts management console, create a notebook instance with a proper AI engine and remote SSH enabled.

3. Use the local IDE to remotely access ModelArts DevEnviron.

4. **Upload data and code to the development environment.**
 - Copy the code to the local IDE, which will automatically synchronize the code to the in-cloud development environment.
 - If the data is less than or equal to 500 MB, directly copy the data to the local IDE.
 - **When creating a training job**, if the volume of data is greater than 500 MB, upload the data to OBS and then to EVS.
5. Upload the training script and dataset to the OBS directory.
6. Submit a training job.
 - Submit a training job in the local IDE.
Use ModelArts SDKs. For details, see [Creating a Training Job](#).
 - Submit a training job on the ModelArts management console. For details, see [Creating a Training Job](#).

5.2 Local IDE (PyCharm)

5.2.1 Connecting to a Notebook Instance Through PyCharm Toolkit

5.2.1.1 PyCharm Toolkit

AI developers use PyCharm tools to develop algorithms or models. Therefore, ModelArts provides PyCharm Toolkit to help AI developers quickly submit locally developed code to a training environment on ModelArts. With PyCharm Toolkit, developers can quickly upload code, submit training jobs, and obtain training logs for local display so that they can better focus on local code development. For details about how to download and install PyCharm Toolkit, see [Installing Through Marketplace](#).

Constraints

- Currently, only PyCharm 2019.2 or later is supported, including the community and professional editions.
- Only PyCharm of the professional edition can be used to access the notebook development environment.
- 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 PyCharm of the Window version.

Available Functions

Table 5-1 Toolkit functions of the latest version

Function	Description	Reference
Remote SSH	The notebook development environment can be accessed through remote SSH.	Connecting to a Notebook Instance Through PyCharm Toolkit
Model training	Code developed locally can be quickly submitted to ModelArts and a training job of the new version is automatically created. During the running of the training job, training logs can be obtained and displayed on a local host.	<ul style="list-style-type: none">• Submitting a Training Job (New Version)• Stopping a Training Job• Viewing Training Logs
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 Using PyCharm

5.2.1.2 Downloading and Installing PyCharm Toolkit

Before using PyCharm Toolkit, install and configure it in PyCharm by following the instructions provided in this section.

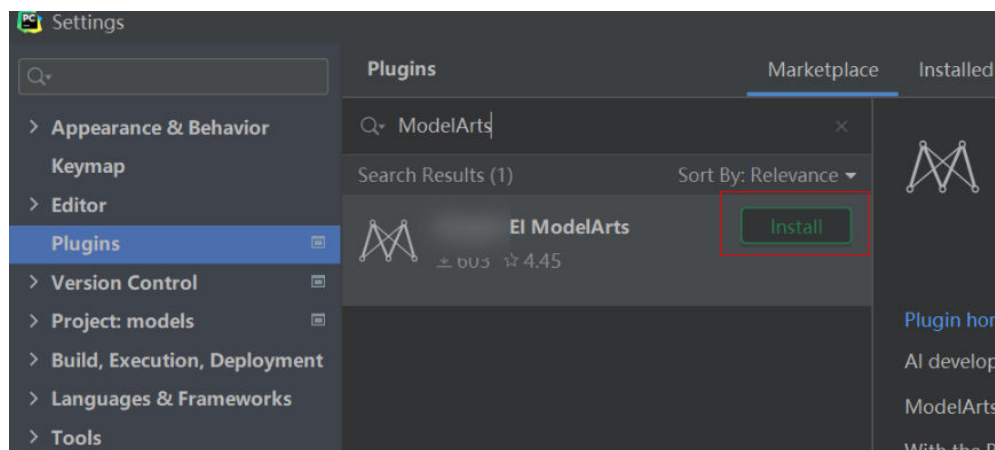
Prerequisites

PyCharm community or professional 2019.2 or later has been installed locally.

- Only PyCharm of the professional edition can be used to access the notebook development environment.
- You can use a community or professional edition of PyCharm Toolkit to submit training jobs. PyCharm Toolkit 2.x can be used to submit only the old version of training jobs, and the latest version of PyCharm Toolkit can be used to submit only the new version of training jobs.

Installing Through Marketplace

In PyCharm, choose **File > Settings > Plugins**, search for **ModelArts** in Marketplace, and click **Install**.

Figure 5-2 Installing through Marketplace**NOTE**

- The version installed in Marketplace is the latest version.
- If ModelArts cannot be found in Marketplace, your network may be restricted. Ensure that you can access the Internet.

5.2.1.3 Connecting to a Notebook Instance Through PyCharm Toolkit

ModelArts provides the PyCharm plug-in PyCharm Toolkit for you to remotely access a notebook instance through SSH, upload code, submit a training job, and obtain training logs for local display.

Prerequisites

PyCharm professional 2019.2 or later has been installed locally. Remote SSH applies only to the PyCharm professional edition. [Download PyCharm](#) and install it.

NOTE

Download PyCharm Professional 2023.2 or an earlier version. The PyCharm Toolkit is not adapt to PyCharm Professional whose version is later than 2023.2.

Step 1 Create 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](#).

Step 2 Download and Install PyCharm Toolkit

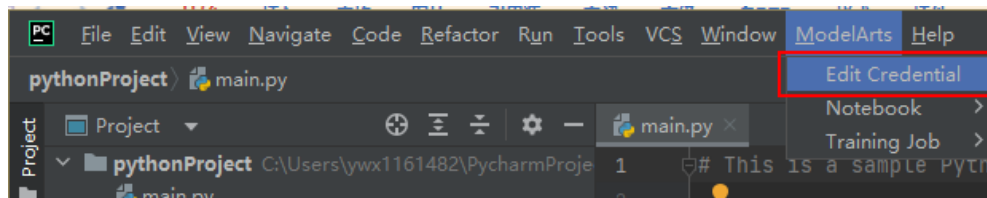
In PyCharm, choose **File > Settings > Plugins**, search for **ModelArts** in Marketplace, and click **Install**. For details, see [Downloading and Installing PyCharm Toolkit](#).

Step 3 Log In to the Plug-in

To use the AK/SK pair for login authentication, perform the following steps:

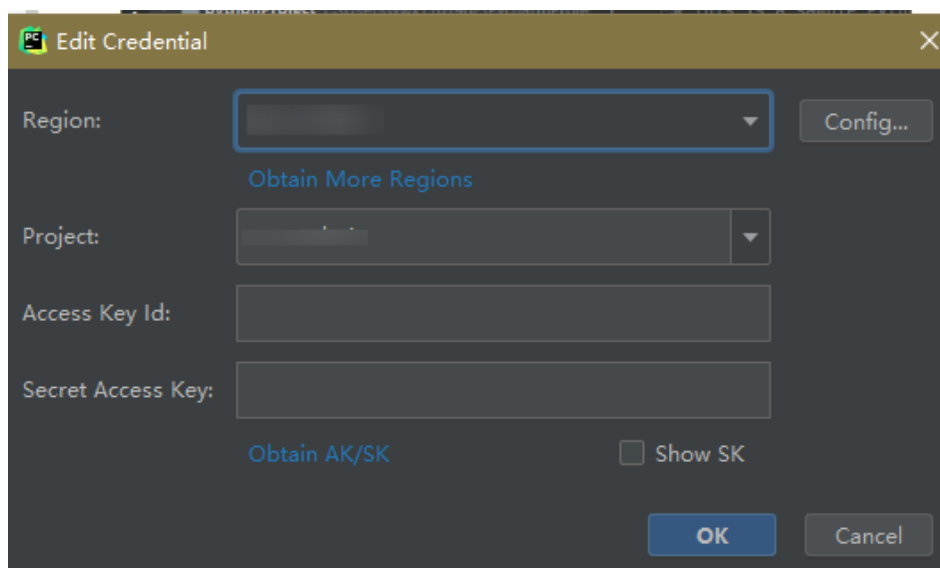
1. Open PyCharm with Toolkit installed. Choose **ModelArts > Edit Credential** from the menu bar.

Figure 5-3 Edit Credential



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 and 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 5-4 Entering the region and access keys



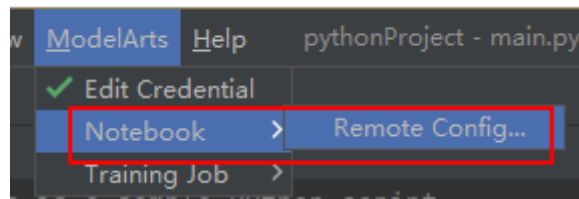
3. View the verification result.
In the **Event Log** area, if information similar to the following is displayed, the access key has been successfully added:

16:01Validate Credential Success: The HUAWEI CLOUDcredential is valid.

Step 4 Automatically Configure PyCharm Toolkit

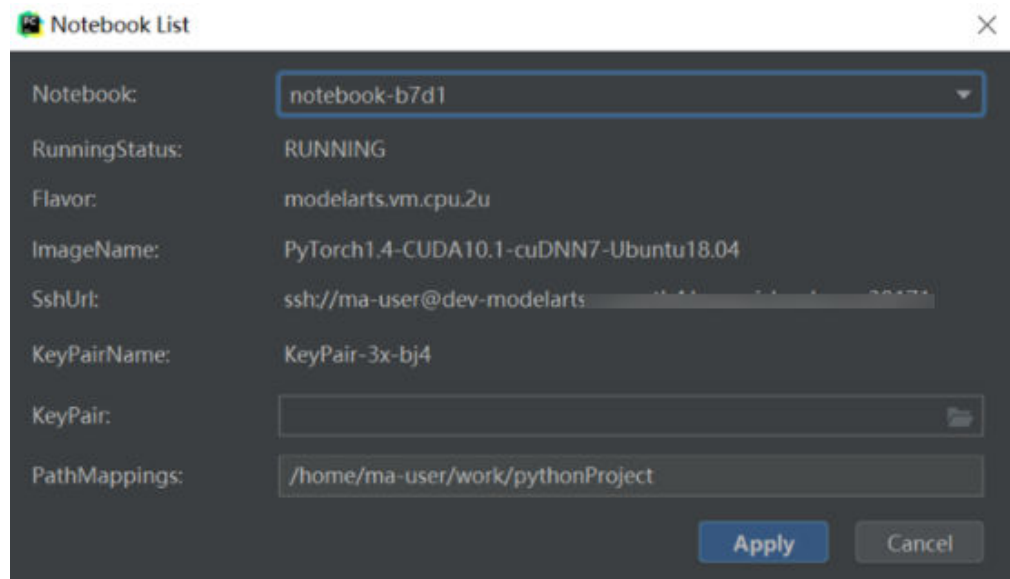
1. In the local PyCharm development environment, choose **ModelArts > Notebook > Remote Config...**, and configure PyCharm Toolkit.

Figure 5-5 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 5-6 Notebook list

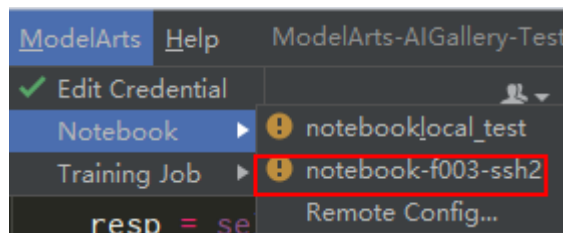


- **KeyPair:** Select the locally stored key pair of the notebook instance for authentication. The key pair created during the notebook instance creation is saved in your browser's default downloads folder.
 - **PathMappings:** Synchronization directory for the local IDE project and notebook, which defaults 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 Access a Notebook Instance Through PyCharm Toolkit

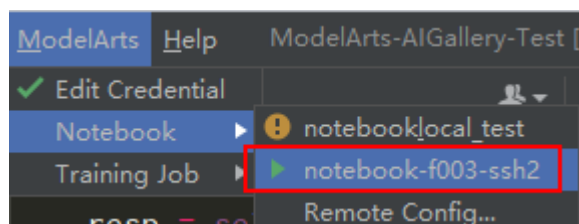
Click the notebook instance name and connect it to the local IDE as prompted. The connection is kept for 4 hours by default.

Figure 5-7 Starting the connection



To interrupt the connection, click the notebook name and disconnect it from the local IDE as prompted.

Figure 5-8 Interrupting the connection



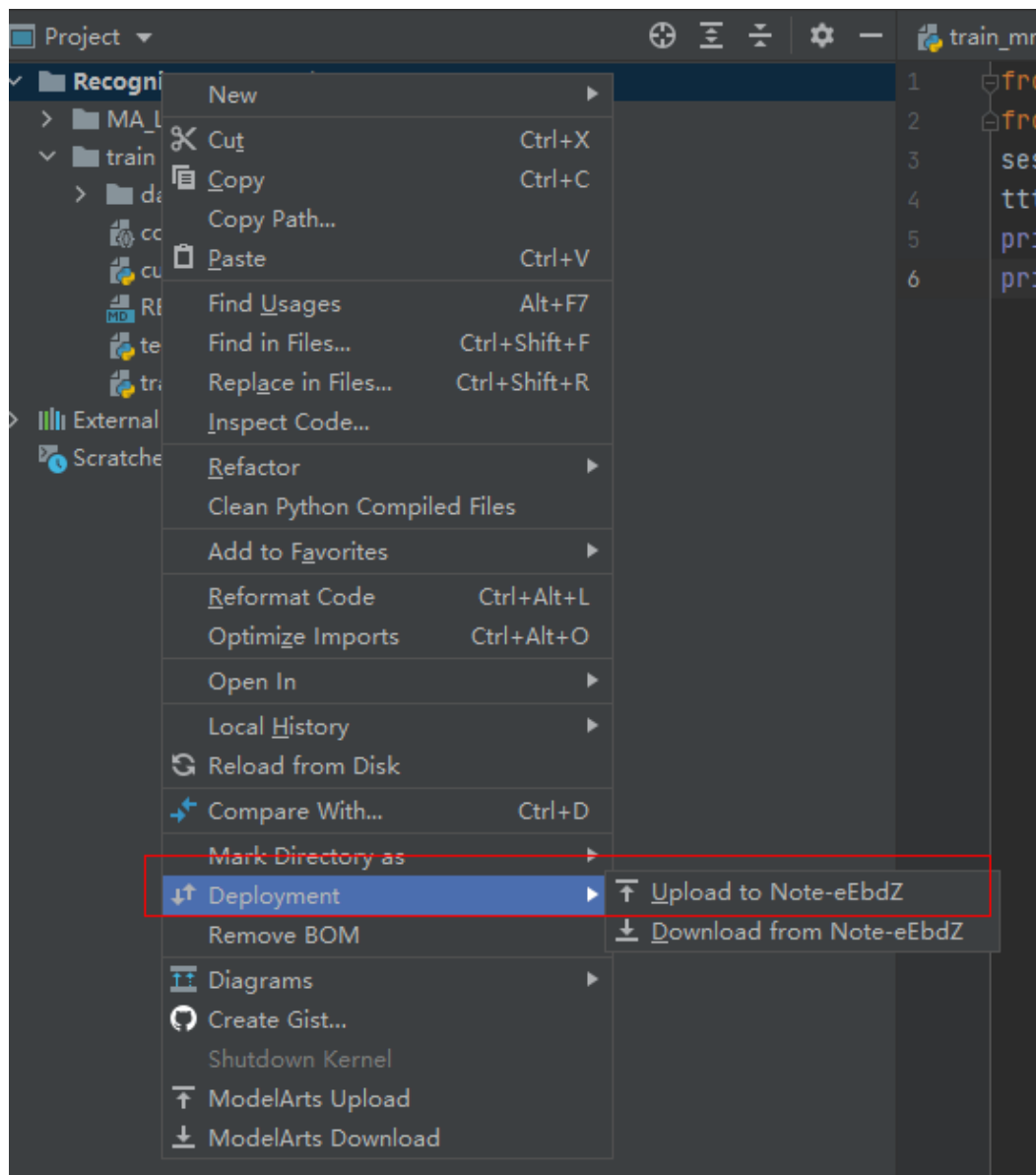
Step 6 Upload Local Files to the Notebook Instance

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 5-9 Synchronizing local data to a notebook instance

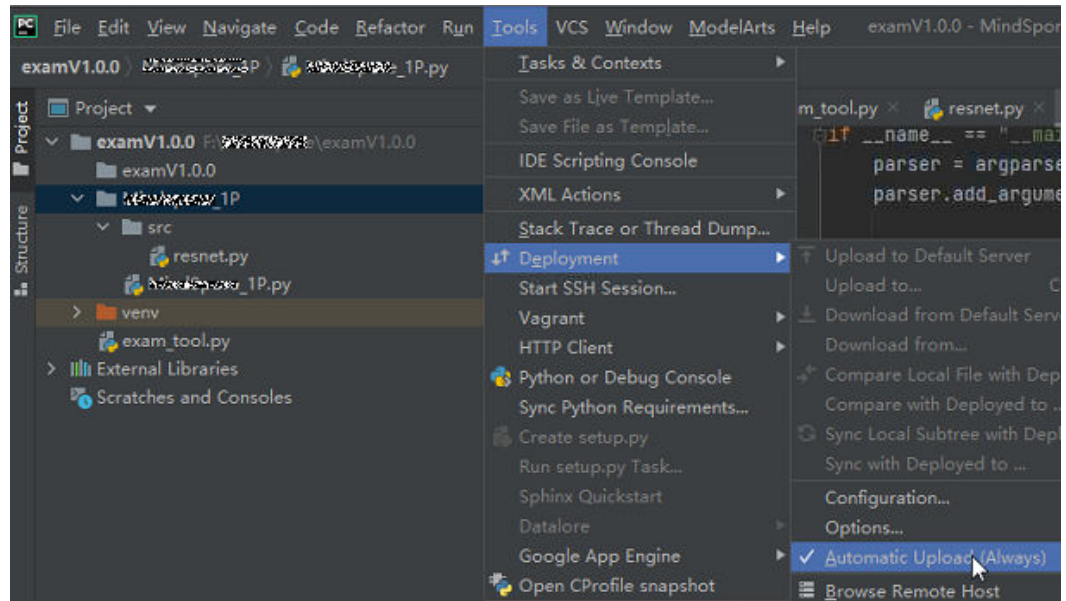


Follow-up synchronization

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.

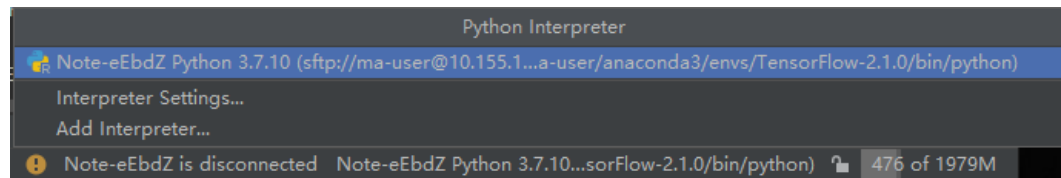
Figure 5-10 Enabling Automatic Upload



Step 7 Remotely Debug the Code

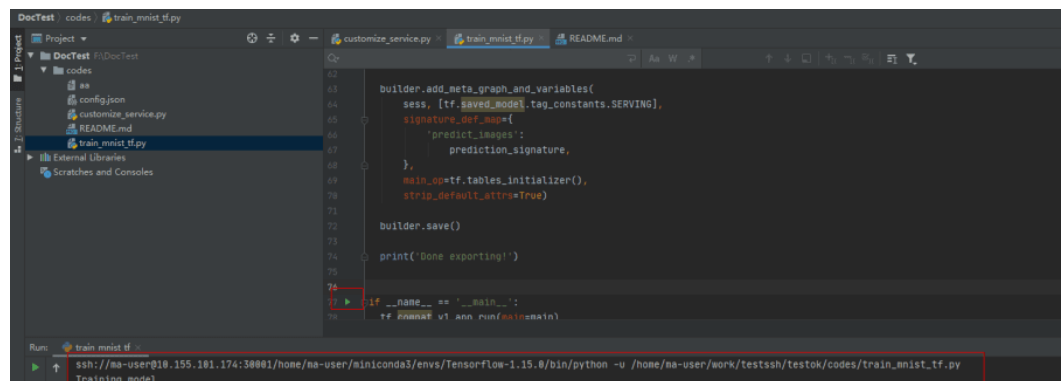
Click **Interpreter** in the lower right corner of the local IDE and select a notebook Python interpreter.

Figure 5-11 Selecting a Python interpreter



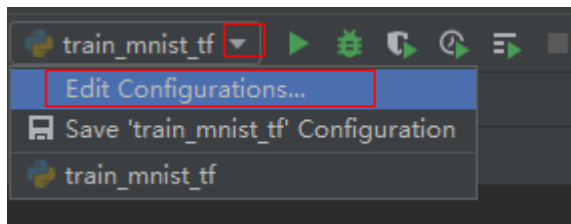
Run the code in the notebook instance. The logs are displayed locally.

Figure 5-12 Runtime logs



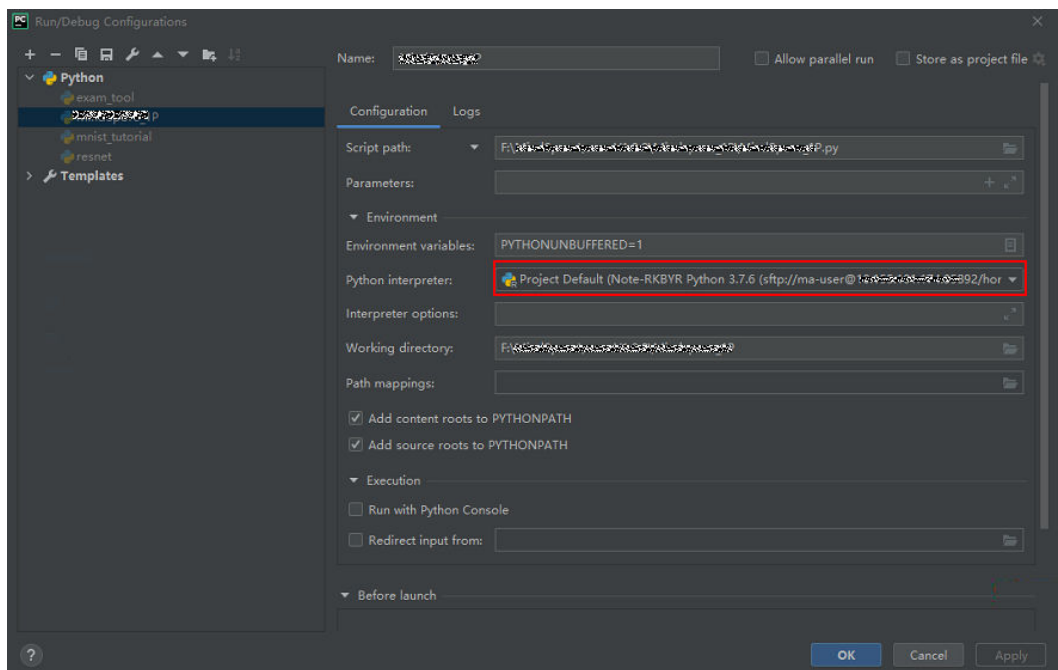
Click **Run/Debug Configurations** in the upper right corner of the local IDE to set runtime parameters.

Figure 5-13 Setting runtime parameters (1)



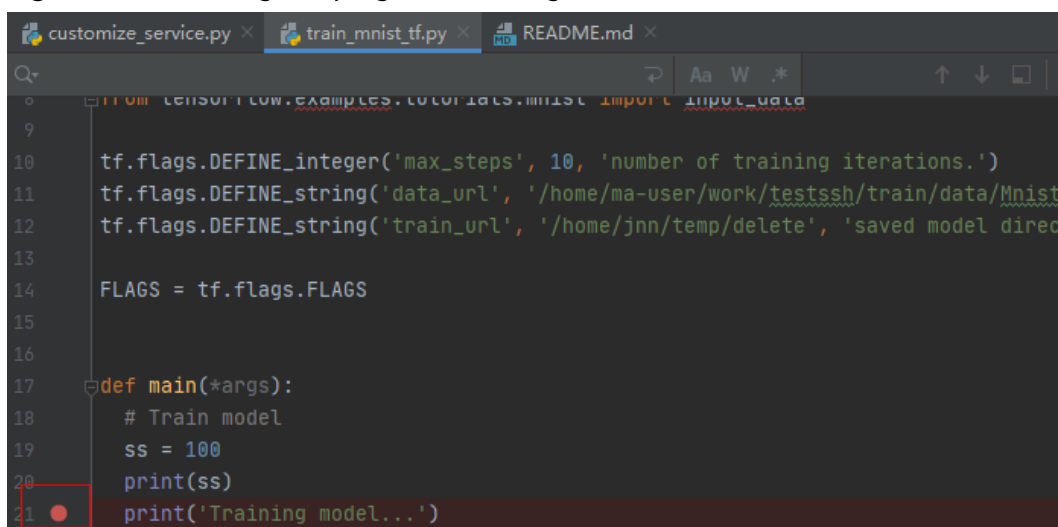
Select the Python interpreter that remotely connects to the target notebook instance.

Figure 5-14 Setting runtime parameters (2)



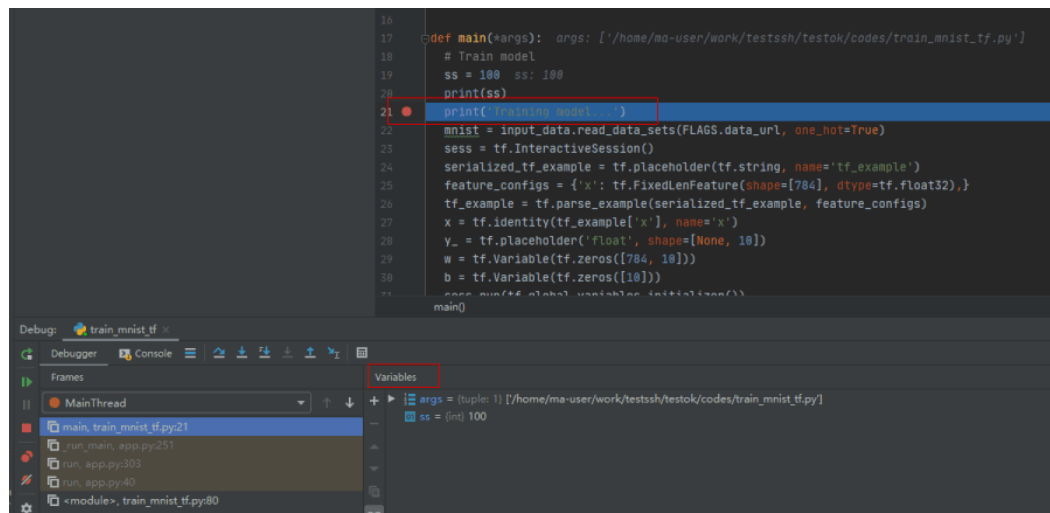
To debug code, set breakpoints and run the program in debug mode.

Figure 5-15 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 5-16 Viewing variable values in debug mode



5.2.2 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](#).
- The address and port number of the development environment are available. To obtain this information, go to the notebook instance details page.

Figure 5-17 Instance details page

Address	ssh://ma-user@dev-modelarts- XXXXXXXXXX .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


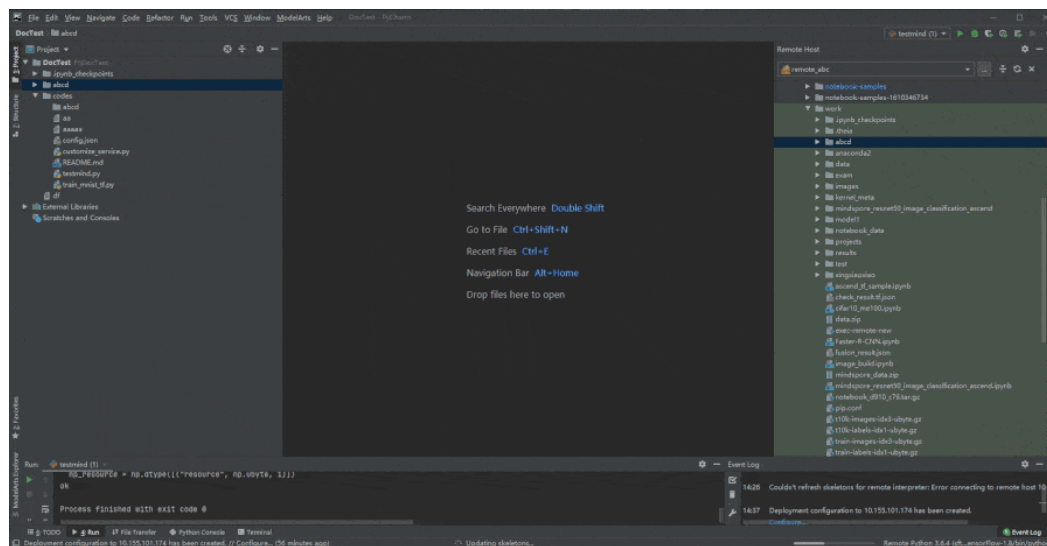
1. In your local PyCharm development environment, choose **File > Settings > Tools > SSH Configurations** and click **+** to add an SSH configuration.
 - **Host:** address for accessing the cloud development environment. Obtain the address on the page providing detailed information of the target notebook 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 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 5-18 Configuring SSH



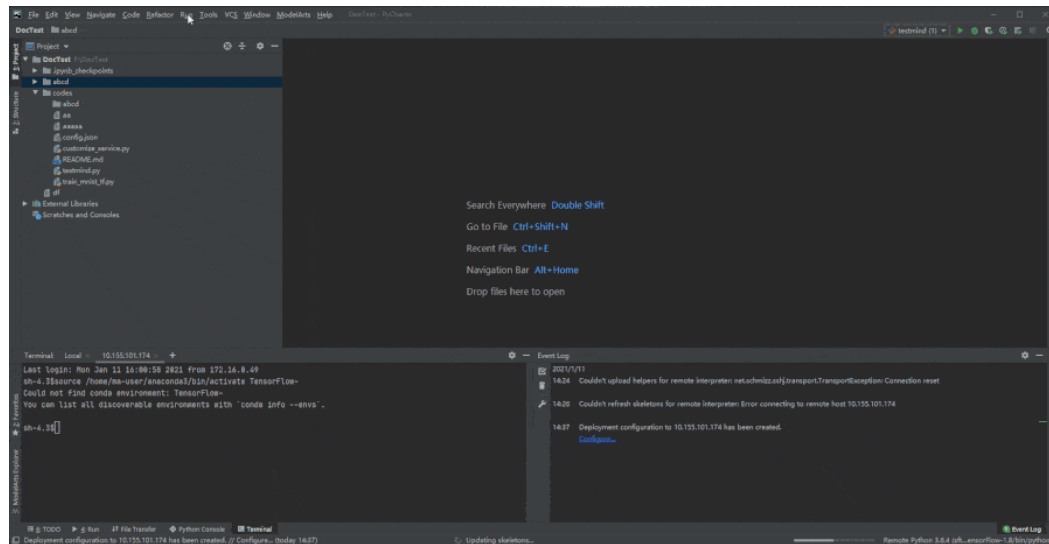
Step 2 Obtain the Path to the Virtual Environment Built in the Development Environment

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 5-19 Obtaining the path to the virtual environment built in the development environment



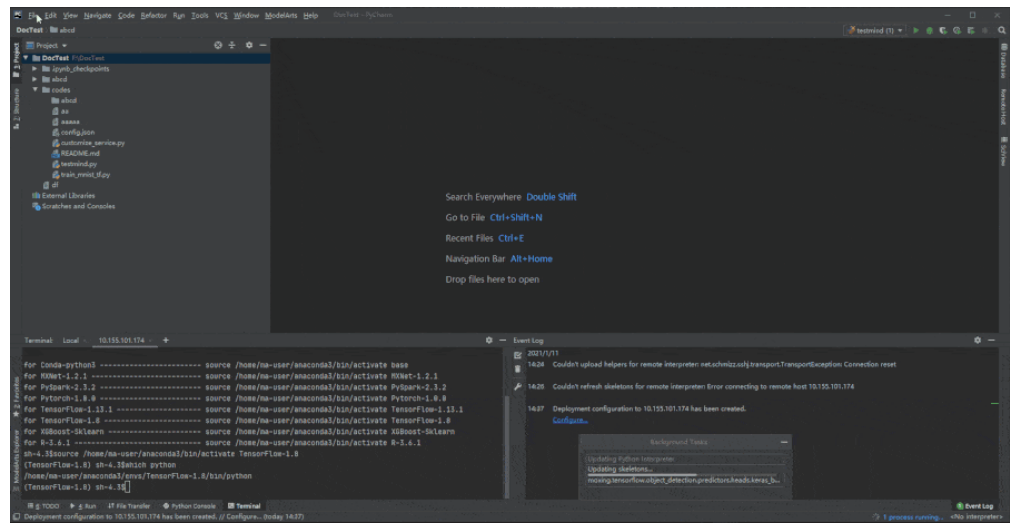
Step 3 Configure a Python Interpreter

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.

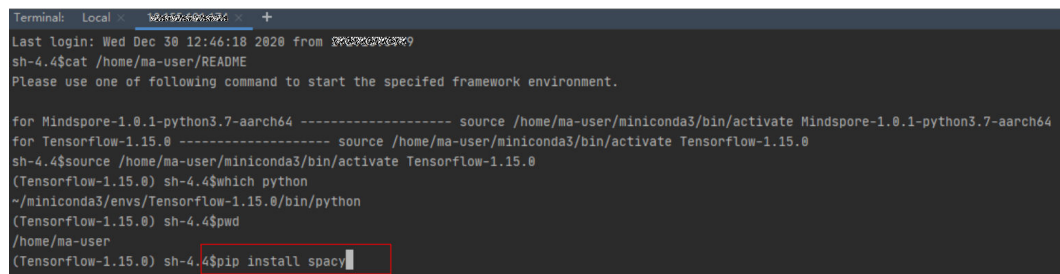
Figure 5-20 Configuring a Python interpreter



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.

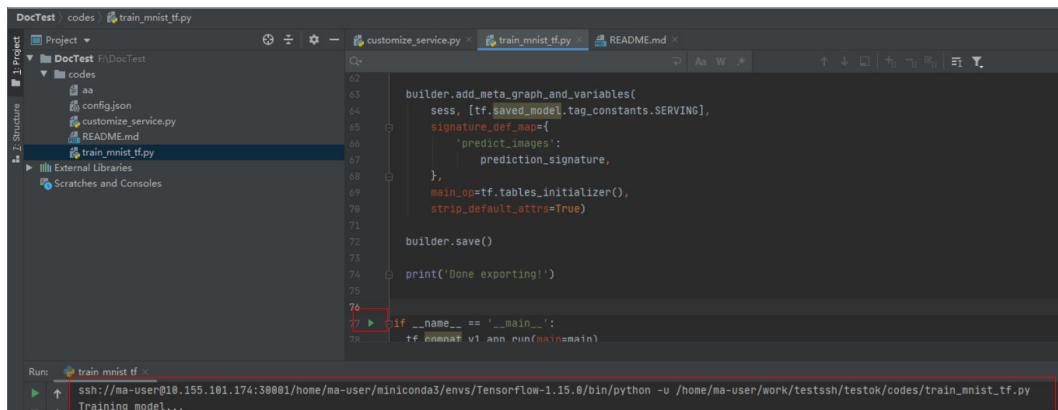


Step 5 Debug Code in the Development Environment

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.

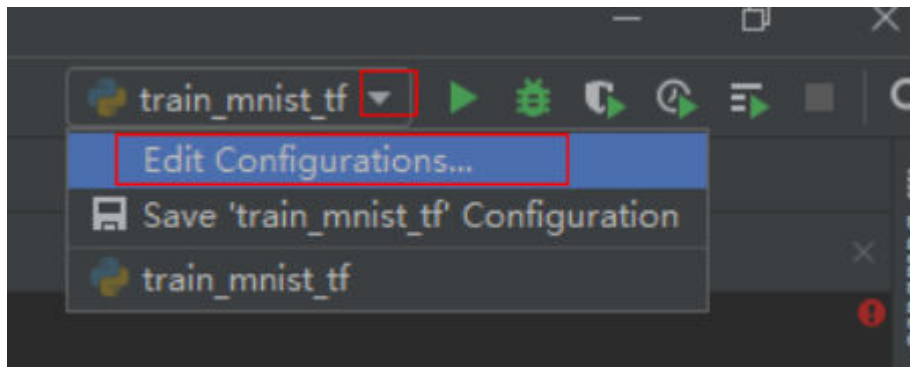
Run the code in the local IDE. The logs can be displayed locally.

Figure 5-21 Debugging code



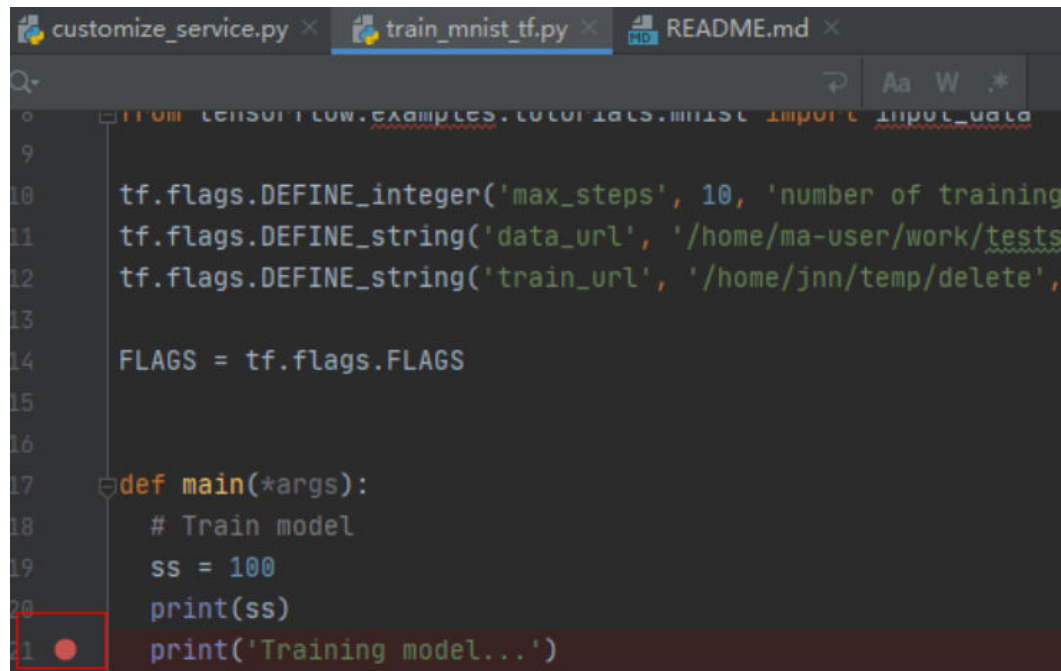
Click **Run/Debug Configurations** in the upper right corner of the local IDE to set runtime parameters.

Figure 5-22 Setting runtime parameters



To debug code, set breakpoints and run the program in debug mode.

Figure 5-23 Code breakpoint



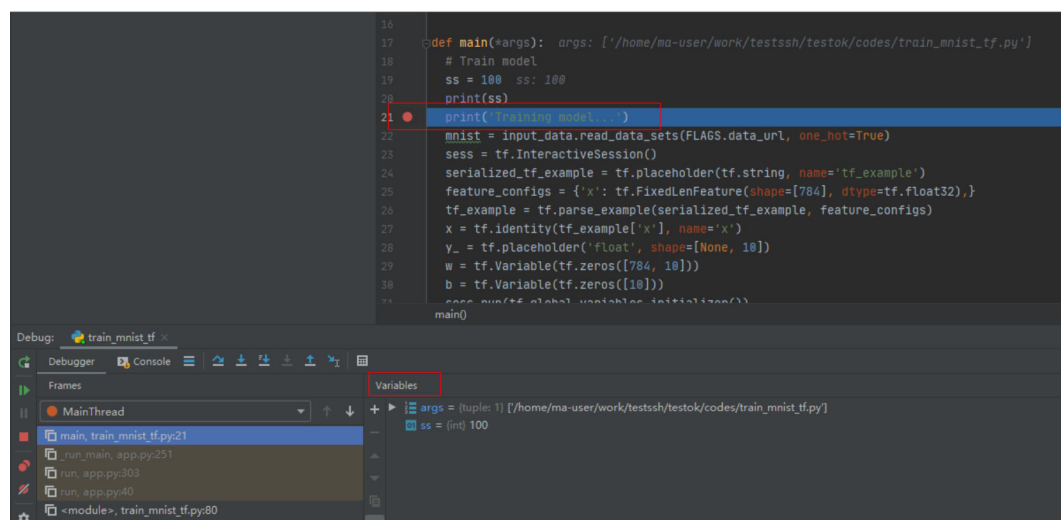
```
customize_service.py x train_mnist_tf.py x README.md x
9
10 tf.flags.DEFINE_integer('max_steps', 10, 'number of training
11 tf.flags.DEFINE_string('data_url', '/home/ma-user/work/testst
12 tf.flags.DEFINE_string('train_url', '/home/jnn/temp/delete',
13
14 FLAGS = tf.flags.FLAGS
15
16
17 def main(*args):
18     # Train model
19     ss = 100
20     print(ss)
21     print('Training model...')
```

Figure 5-24 Debugging in debug mode



In debug mode, the code execution is suspended in the specified line, and you can obtain variable values.

Figure 5-25 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 Upload Local Files to the Notebook Instance](#).

5.2.3 Submitting a Training Job Using PyCharm Toolkit

5.2.3.1 Submitting a Training Job (New Version)

You can use PyCharm Toolkit of the latest version to quickly submit the locally developed training code to ModelArts for training.

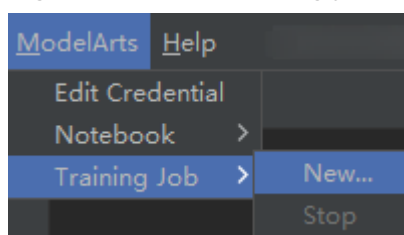
Prerequisites

- A training code project exists in the local PyCharm.
- You have created a bucket and folders in OBS for storing datasets and trained models. Data used by the training job has been uploaded to OBS.
- The credential has been configured. For details, see [Using Access Keys for Login](#).
- PyCharm Toolkit of the latest version is available for submitting a training job of the new version only.

Configuring Training Job Parameters

1. In PyCharm, open the training code project and training boot file, and choose **ModelArts > Training Job > New...** on the menu bar.

Figure 5-26 Edit training job configuration



2. In the displayed dialog box, configure the training job parameters. For details, see [Table 5-2](#).

Table 5-2 Training job parameters

Parameter	Description
Job Name	Name of a training job The system automatically generates a name. You can rename it based on the following naming rules: <ul style="list-style-type: none">• The name contains 1 to 64 characters.• Letters, digits, hyphens (-), and underscores (_) are allowed.
Job Description	Brief description of a training job
Algorithm Source	Source of the training algorithm. The options are Frequently-used and Custom . Frequently-used refers to the frequently-used AI engines supported by ModelArts Training Management. If the AI engine you use is not in the supported list, you are advised to create a training job using a custom image.
AI Engine	Select the AI engine and the version used in code. The supported AI engines are the same as the frequently-used frameworks supported by training jobs on the ModelArts management console.
Boot File Path	Training boot file. The selected boot file must be a file in the current PyCharm training project. This parameter is displayed if Algorithm Source is set to Frequently-used .
Code Directory	Training code directory. The system automatically sets this parameter to the directory where the training boot file is located. You can change the parameter value to a directory that is in the current project and contains the boot file. If the algorithm source is a custom image and the training code has been built in the image, this parameter can be left blank.
Image Path(optional)	URL of the SWR image
Boot Command	Command for starting a training job, for example, bash /home/work/run_train.sh python {Python boot file and parameters} . This parameter is displayed if Algorithm Source is set to Custom . If the command does not contain the --data_url or --train_url parameter, the tool automatically adds the two parameters to the end of the command when submitting the training job. The two parameters correspond to the OBS path for storing training data and the OBS path for storing training output, respectively.

Parameter	Description
Data OBS Path	OBS path for storing training data, for example, /test-modelarts2/mnist/dataset-mnist/ , in which test-modelarts2 indicates a bucket name.
Training OBS Path	OBS path. A directory is automatically created in the path for storing a trained model and training logs.
Running Parameters	Running parameters. If you want to add some running parameters to your code, add them here. Separate multiple running parameters with semicolons (;), for example, key1=value1;key2=value2 . This parameter can be left blank.
Specifications	Type of resources used for training. Currently, public resource pools and dedicated resource pools are supported. Dedicated resource pool specifications are identified by Dedicated Resource Pool . Dedicated resource pool specifications are displayed only for users who have purchased dedicated resource pools.
Compute Nodes	Number of compute nodes. If this parameter is set to 1 , the system runs in standalone mode. If this parameter is set to a value greater than 1, the distributed computing mode is used at the background.
Available/Total Nodes	When Specifications is set to a dedicated resource pool, the number of available nodes and the total number of nodes are displayed. The value of Compute Nodes cannot exceed the number of available nodes.

Figure 5-27 Configuring training job parameter (public resource pool)

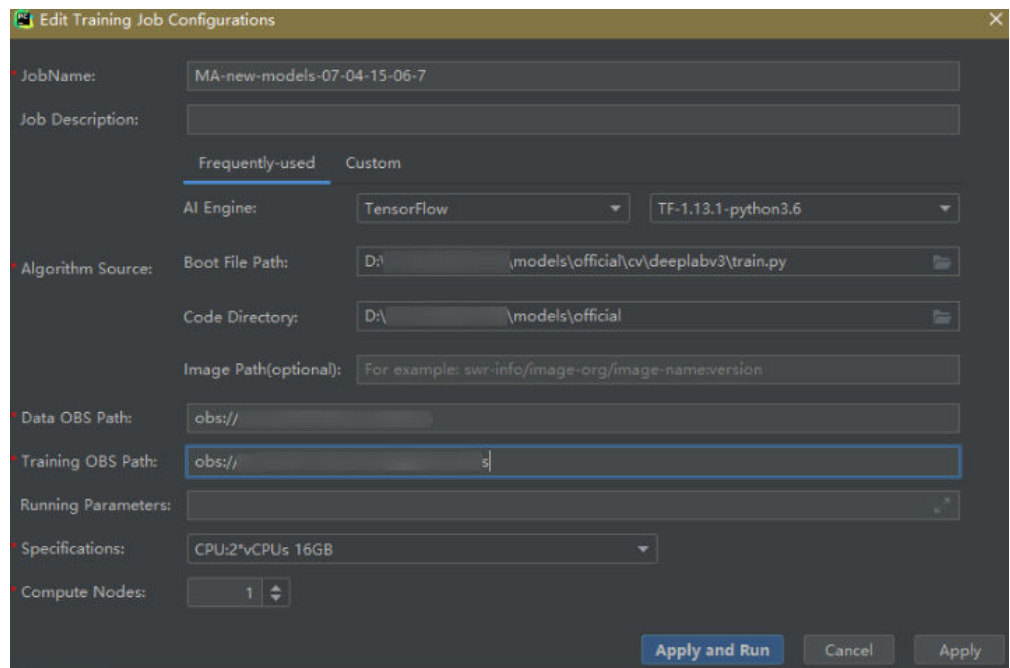


Figure 5-28 Configuring training job parameter (dedicated resource pool)

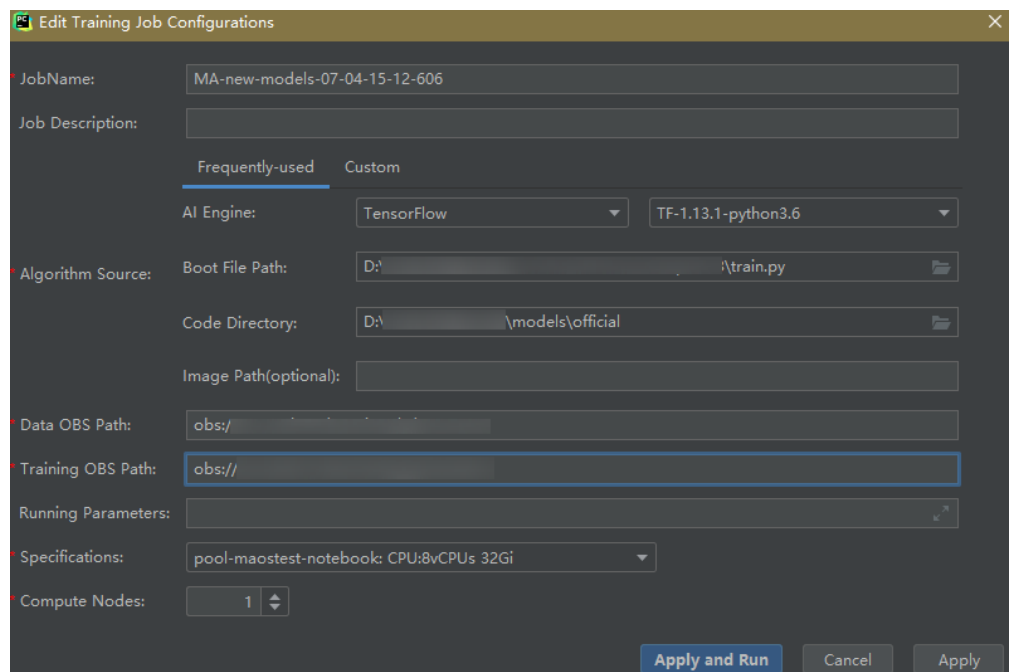
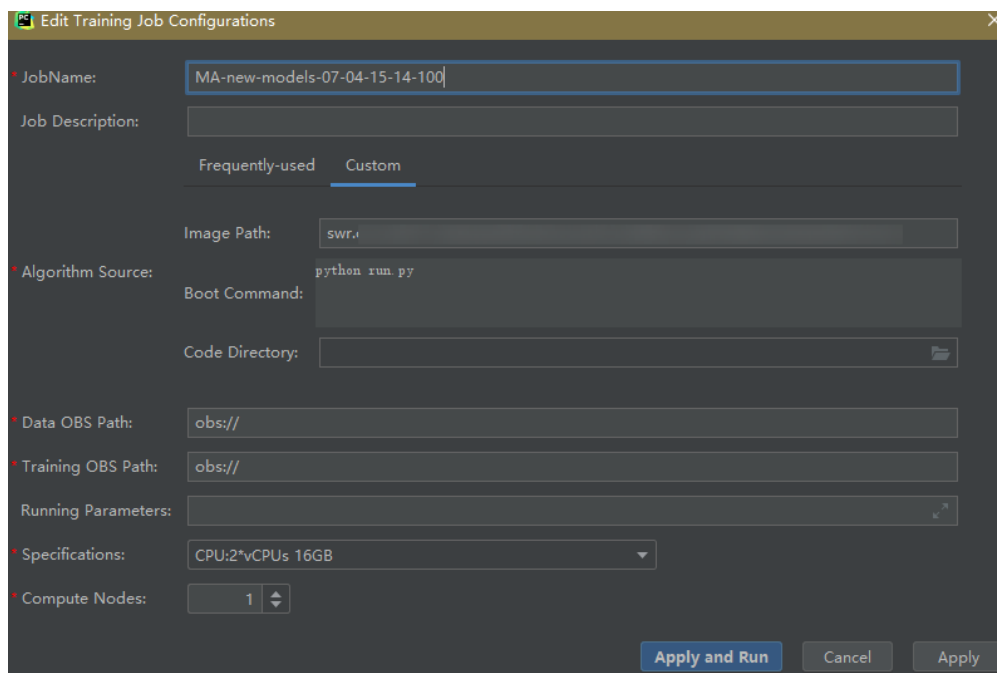


Figure 5-29 Configuring training job parameter (custom image)

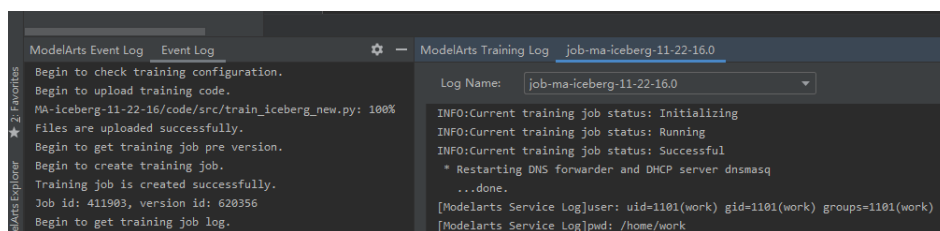


3. After setting the parameters, click **Apply and Run**. Then, local code is automatically uploaded to the cloud and training is started. The training job running status is displayed in the **Training Log** area in real time. If information similar to **Current training job status: Successful** is displayed in the training log, the training job has been successfully executed.

NOTE

- After you click **Apply and Run**, the system automatically executes the training job. To stop the training job, choose **ModelArts > Training Job > Stop** on the menu bar.
- If you click **Apply**, the job is not started directly, and the training job settings are saved instead. To start the job, click **Apply and Run**.

Figure 5-30 Training log example

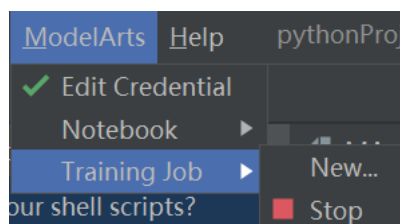


5.2.3.2 Stopping a Training Job

You can stop a running training job.

Stopping a Job

When a training job is running, choose **ModelArts > Training Job > Stop** on the PyCharm menu bar to stop the job.

Figure 5-31 Stopping a job

5.2.3.3 Viewing Training Logs

This section describes how to view training job logs.

Viewing Training Logs in OBS

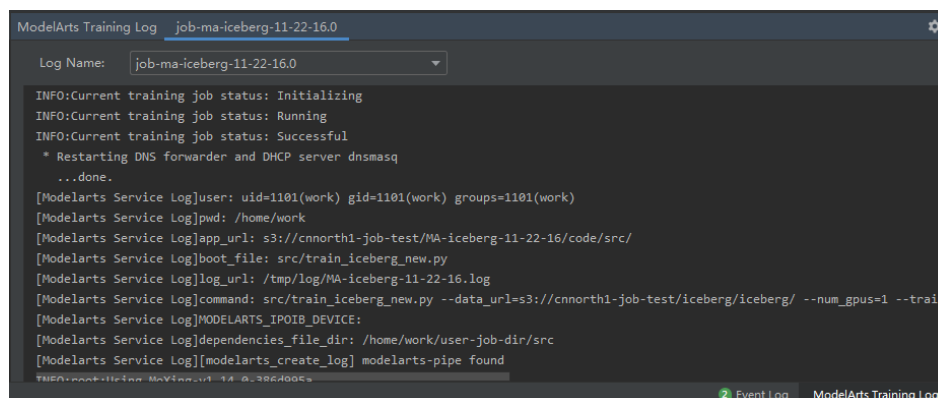
When you submit a training job, the system automatically creates a folder with the same name as the training job in the configured OBS path to store the model, logs, and code outputted after training is complete.

For example, when the **train-job-01** job is submitted, a folder named **train-job-01** is created in the **test-modelarts2** bucket. In this folder, three sub-folders (**output**, **log**, and **code**) are created to store the outputted model, logs, and training code, respectively. Sub-folders will be created in the **output** folder based on your training job version. The following is an example of the folder structure:

```
test-modelarts2
|---train-job-01
|   |---output
|   |---log
|   |---code
```

Viewing Training Logs in Toolkit

In PyCharm, click **ModelArts Training Log** in the lower right corner of the page. The training logs are displayed.

Figure 5-32 Viewing Training Logs

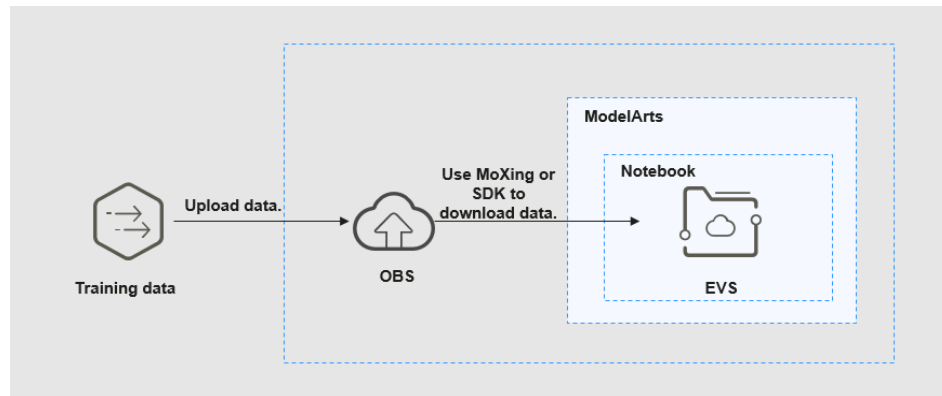
5.2.4 Uploading Data to a Notebook Instance Using PyCharm

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 the code to OBS and then to the notebook instance.

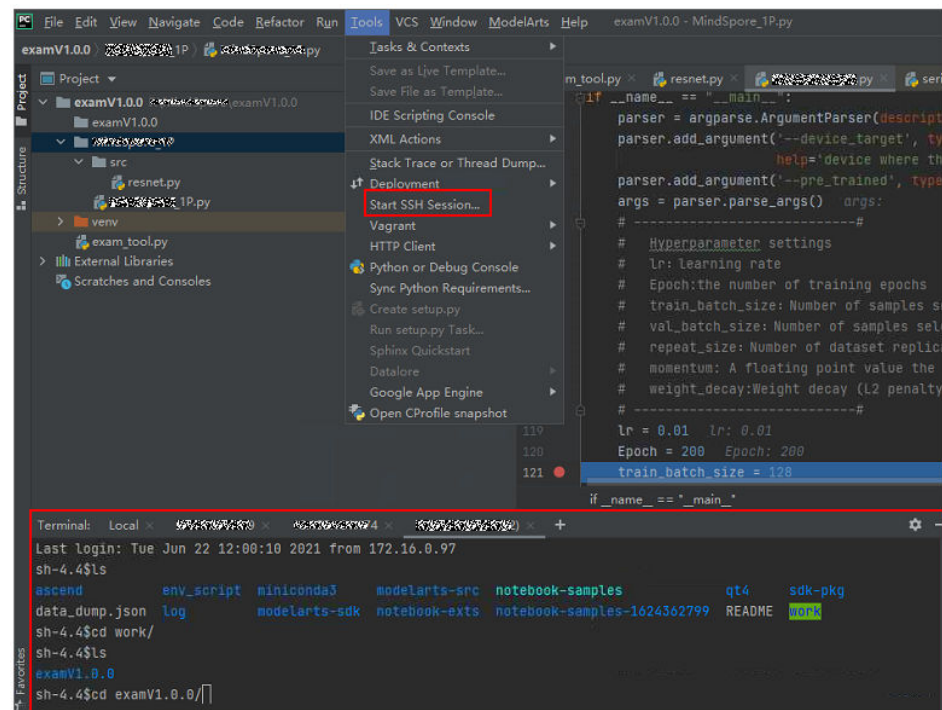
1. Upload data to OBS. For details, see [Uploading an Object](#).
2. Call the `mox.file.copy_parallel` MoXing API provided by ModelArts in the terminal of the local IDE to transfer data from OBS to the notebook instance.

Figure 5-33 Uploading data to a notebook Instance through OBS



The following shows how to enable terminal in PyCharm (the operations in VS Code are similar).

Figure 5-34 Enabling the terminal in PyCharm



The following shows how to use MoXing in the terminal of the local IDE to download files from OBS to a development environment:

```
# 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')
```

5.3 Local IDE (VS Code)

5.3.1 Connecting to a Notebook Instance Through VS Code

After creating a notebook instance with remote SSH enabled, you can use VS Code to access the development environment in either of the following ways:

- **Connecting to a Notebook Instance Through VS Code Toolkit** (Recommended)
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.

5.3.2 Installing VS Code

Download URL:

- Download address: https://code.visualstudio.com/updates/v1_85

Figure 5-35 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.

5.3.3 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.

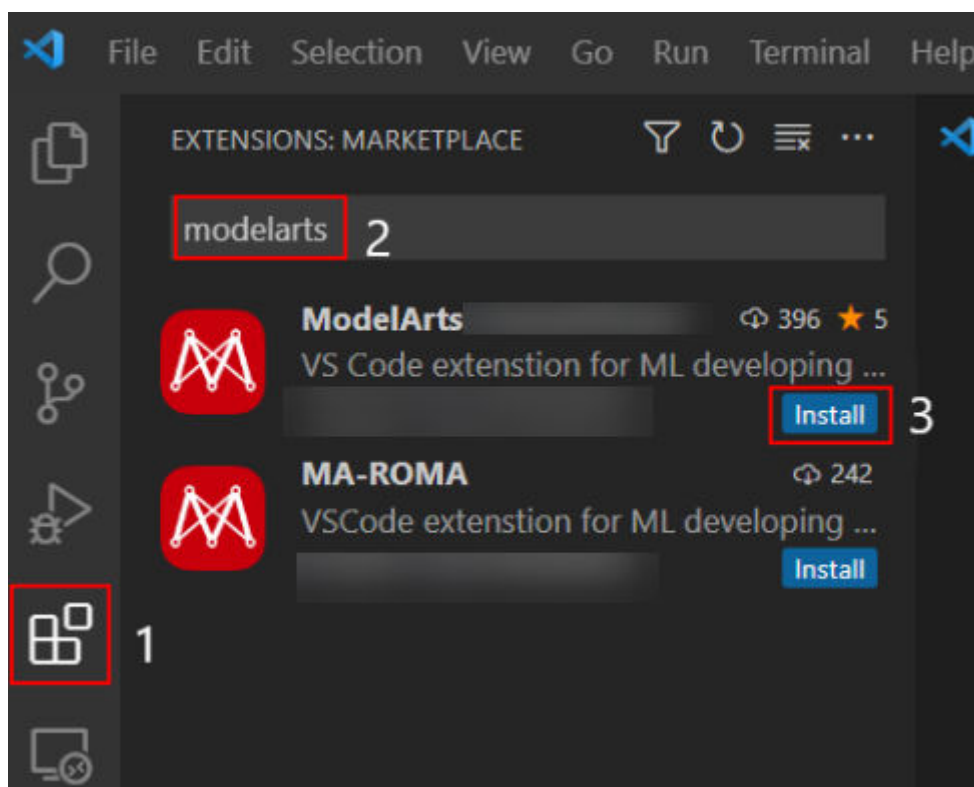
Prerequisites

You have downloaded and installed VS Code. For details, see [Installing VS Code](#).

Step 1 Install the VS Code Plug-in

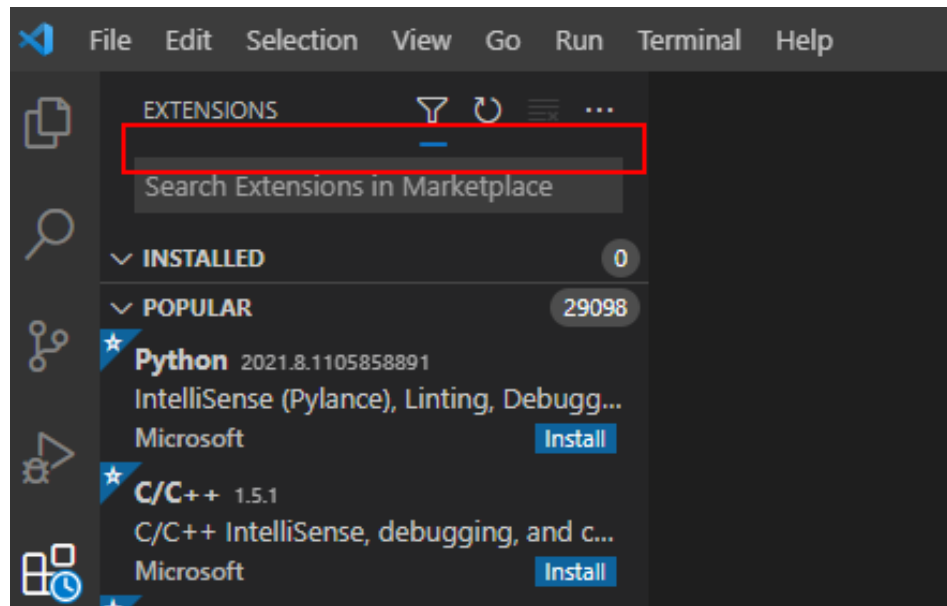
1. Search for **ModelArts-HuaweiCloud** in the **EXTENSIONS** text box and click **Install**.

Figure 5-36 Installing the VS Code plug-in



2. Wait for about 1 to 2 minutes.

Figure 5-37 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 5-38 Installation completion message

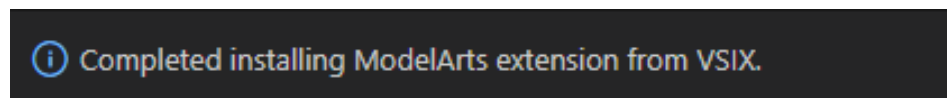
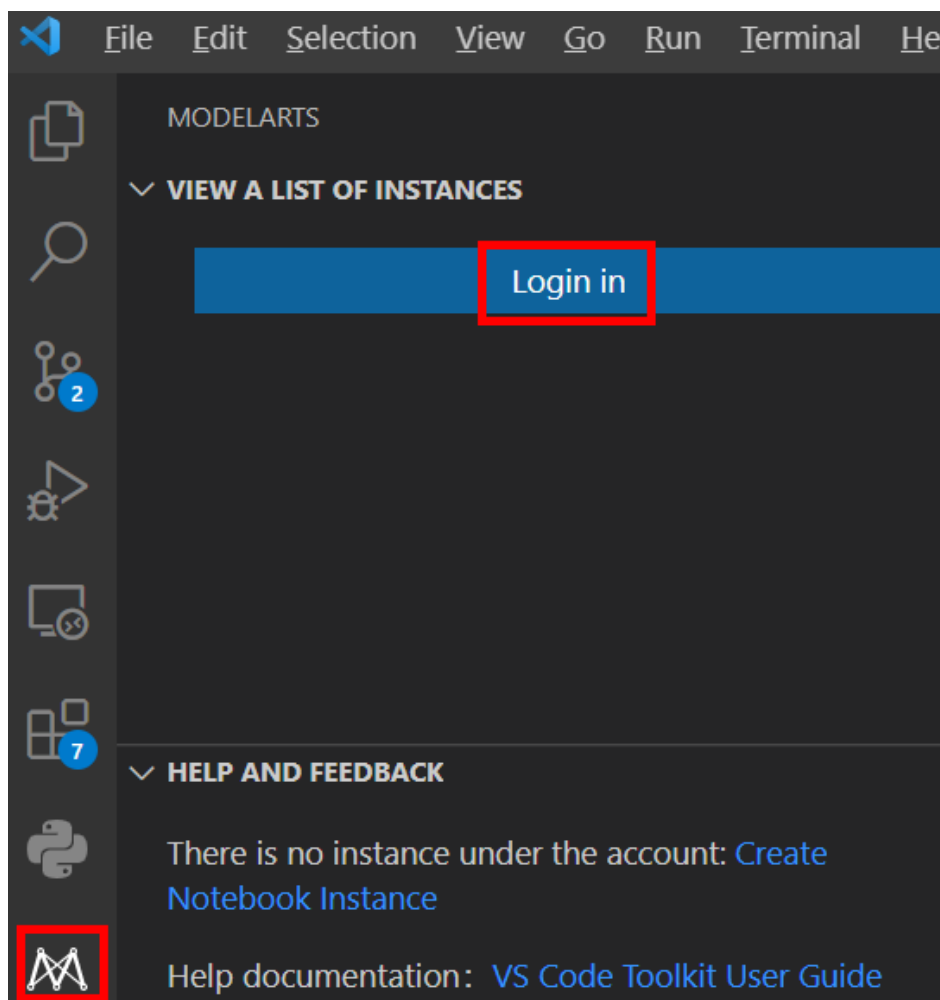
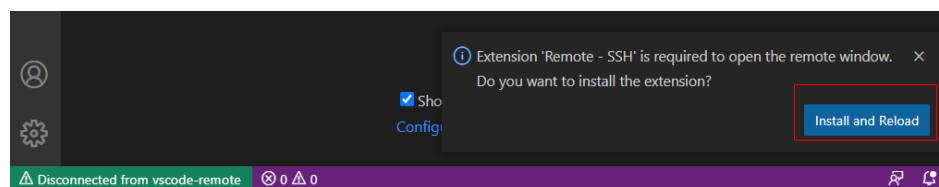


Figure 5-39 Installation completed



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 5-40 Reconnecting remote SSH



Step 2 Log In to the VS Code Plug-in


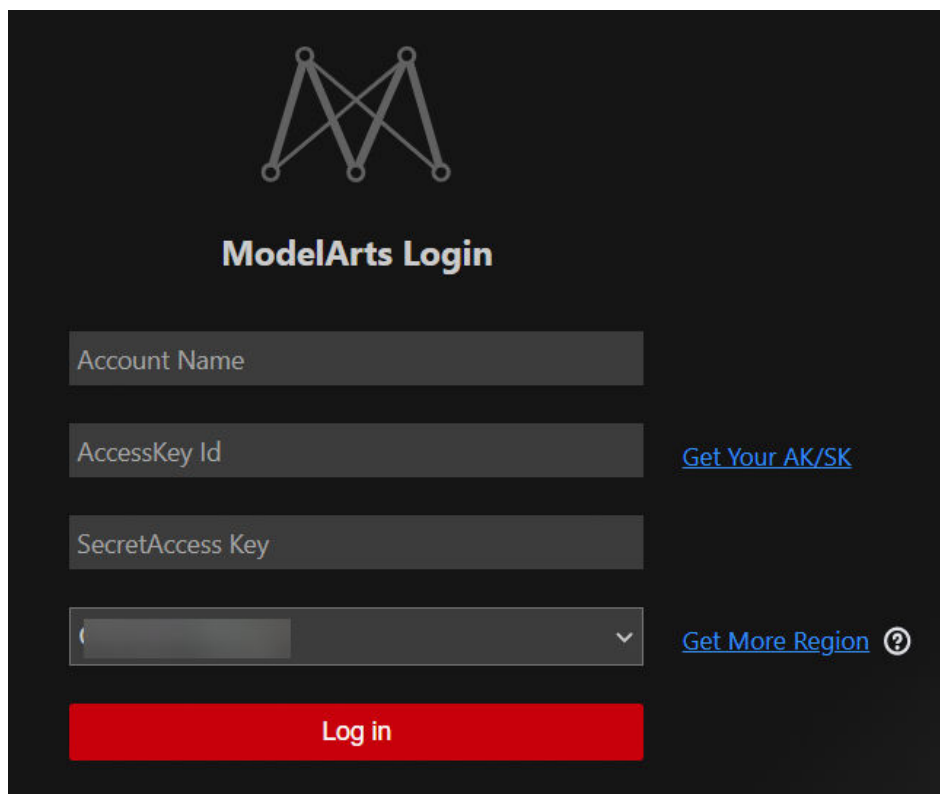
1. In the local VS Code development environment, click  and **User Settings**, and configure the login information.

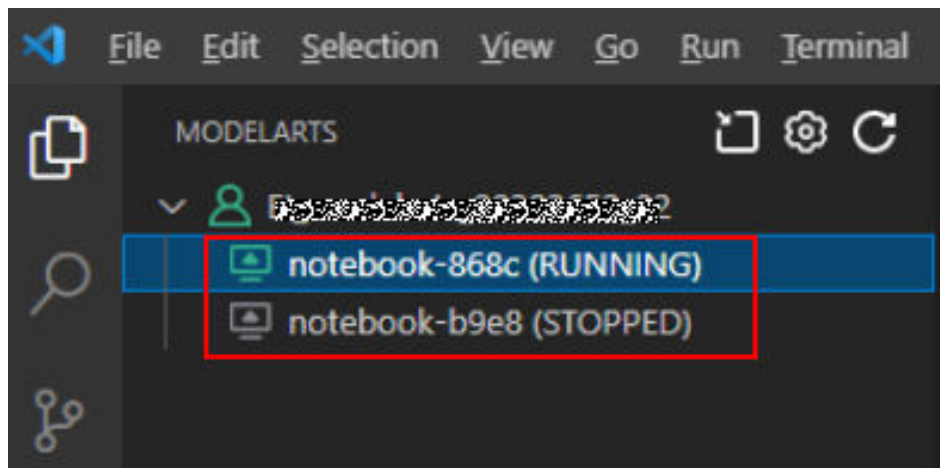
Figure 5-41 Logging in to the plug-in



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**.
 - **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.

Figure 5-42 Login succeeded



Step 3 Create a Notebook Instance

CAUTION

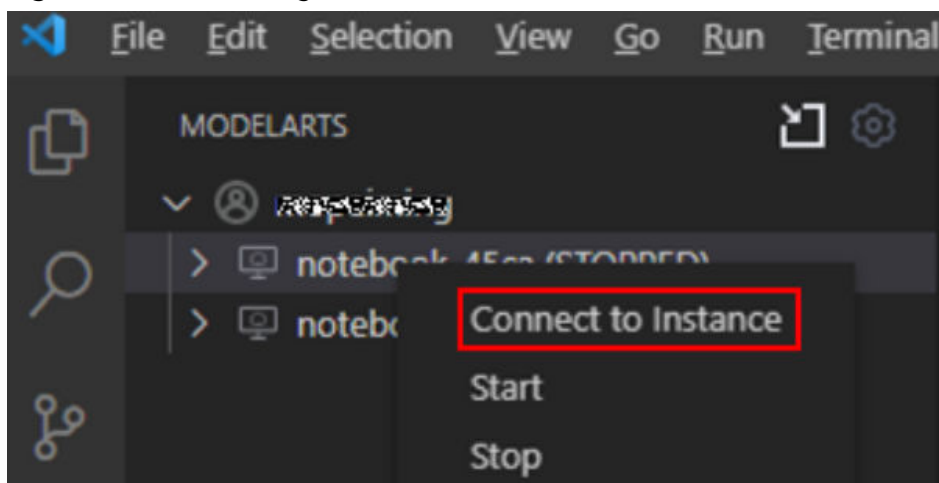
- Create a notebook instance with remote SSH enabled, and download the key file to either of the following directories based on your OS:
Windows: **C:\Users\{{user}}**
macOS or Linux: **Users/{{user}}**
- A key pair is automatically downloaded after you create it. Securely store your key pair. If an existing key pair is lost, create a new one.

Create a notebook instance with remote SSH enabled. For details, see [Creating a Notebook Instance](#).

Step 4 Access the Notebook Instance

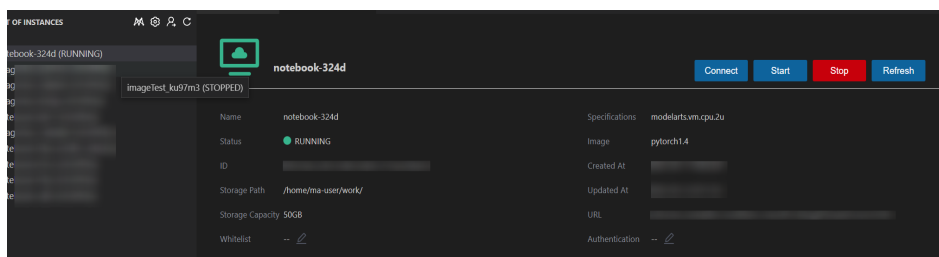
1. In the local VS Code development environment, right-click the instance name and choose **Connect to Instance** from the shortcut menu to start and connect to the notebook instance.
The notebook instance can either be running or stopped. If it is stopped, the VS Code plug-in starts the instance and then connects to it.

Figure 5-43 Connecting to a notebook instance



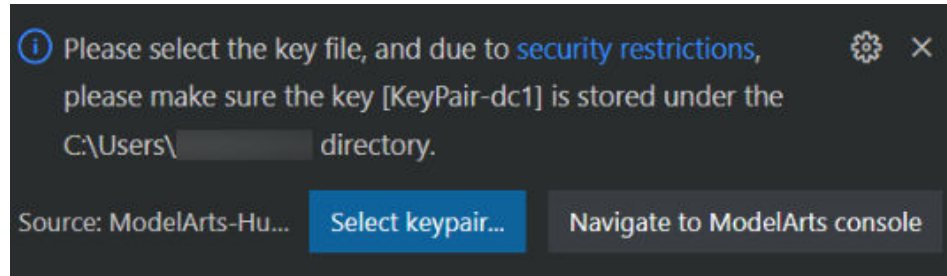
Alternatively, click the instance name. On the instance details page, click **Connect**. Then, the system automatically starts and connects to the notebook instance.

Figure 5-44 Viewing details about a notebook instance



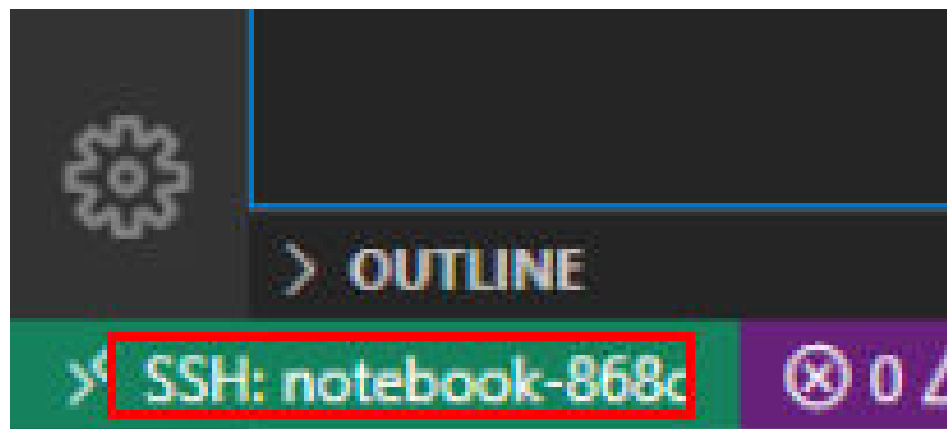
2. When you connect to a notebook instance for the first time, the system prompts you in the lower right corner to configure the key file. In this case, select the local .pem key file and click **OK**.

Figure 5-45 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.

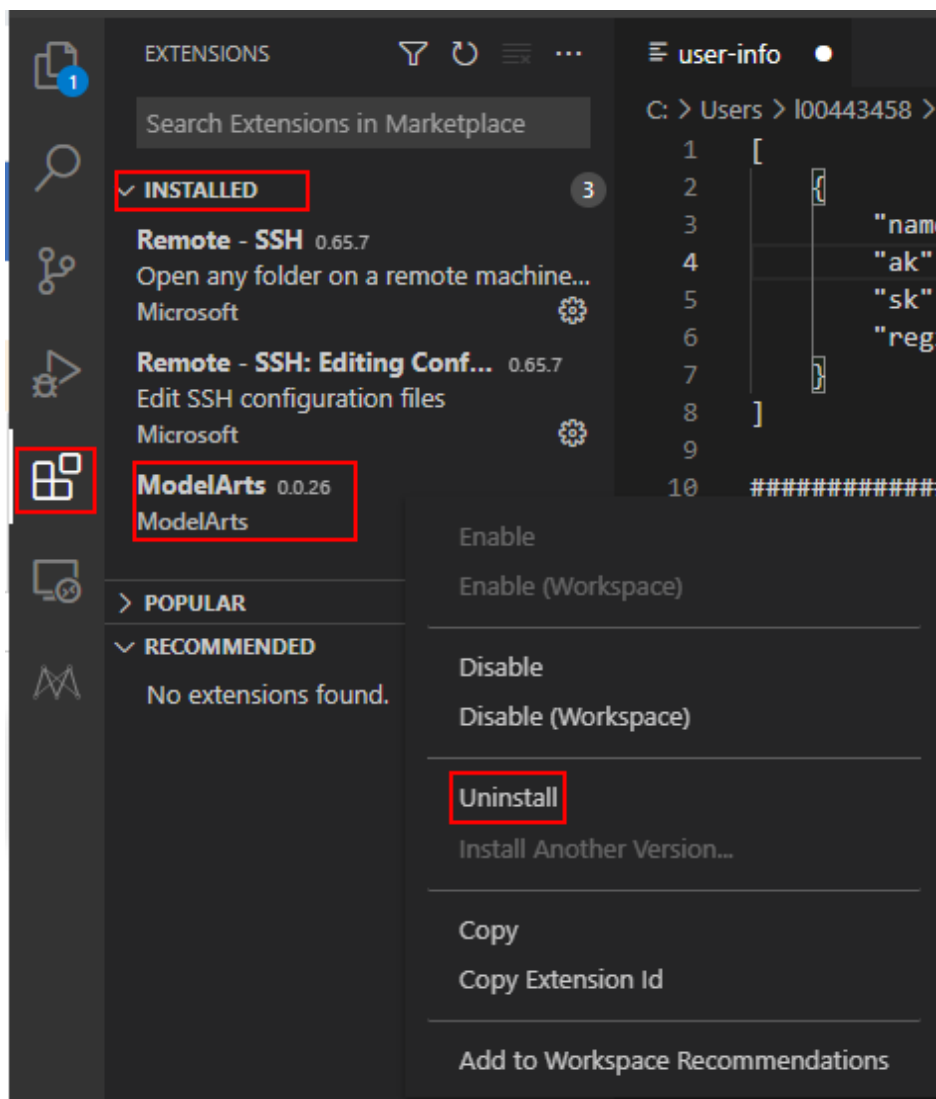
Figure 5-46 Connection succeeded



Related Operations

For details about uninstalling the VS Code plug-in, see [Figure 5-47](#).

Figure 5-47 Uninstalling the VS Code plug-in



5.3.4 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 [Installing 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](#).

- The address and port number of the development environment are available. To obtain the information, go to the notebook instance details page.

Figure 5-48 Instance details page



- 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


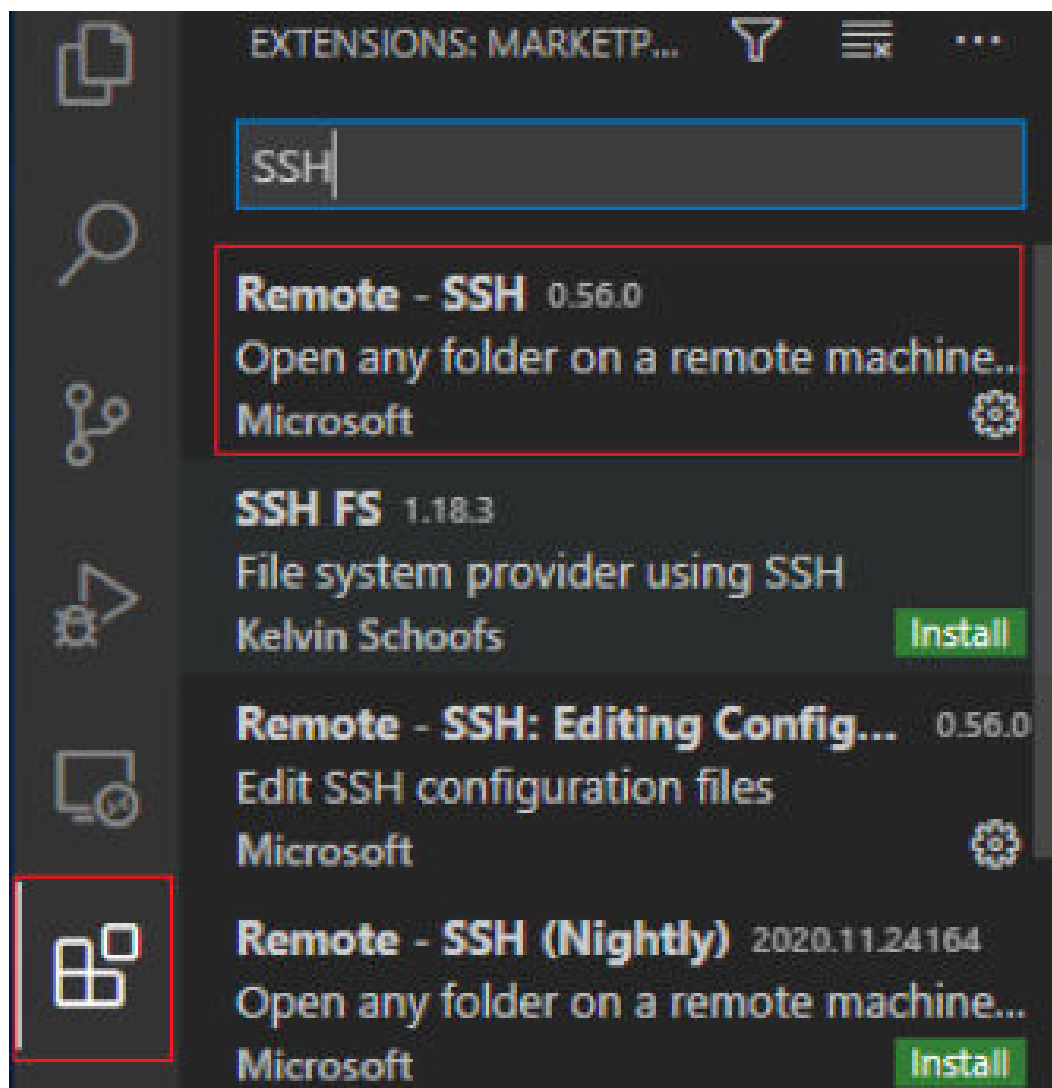
In the local VS Code development environment, click  , enter **SSH** in the search box, and click **install** of the Remote-SSH plug-in to install the plug-in.

Figure 5-49 Adding the Remote-SSH plug-in



Step 2 Configure SSH



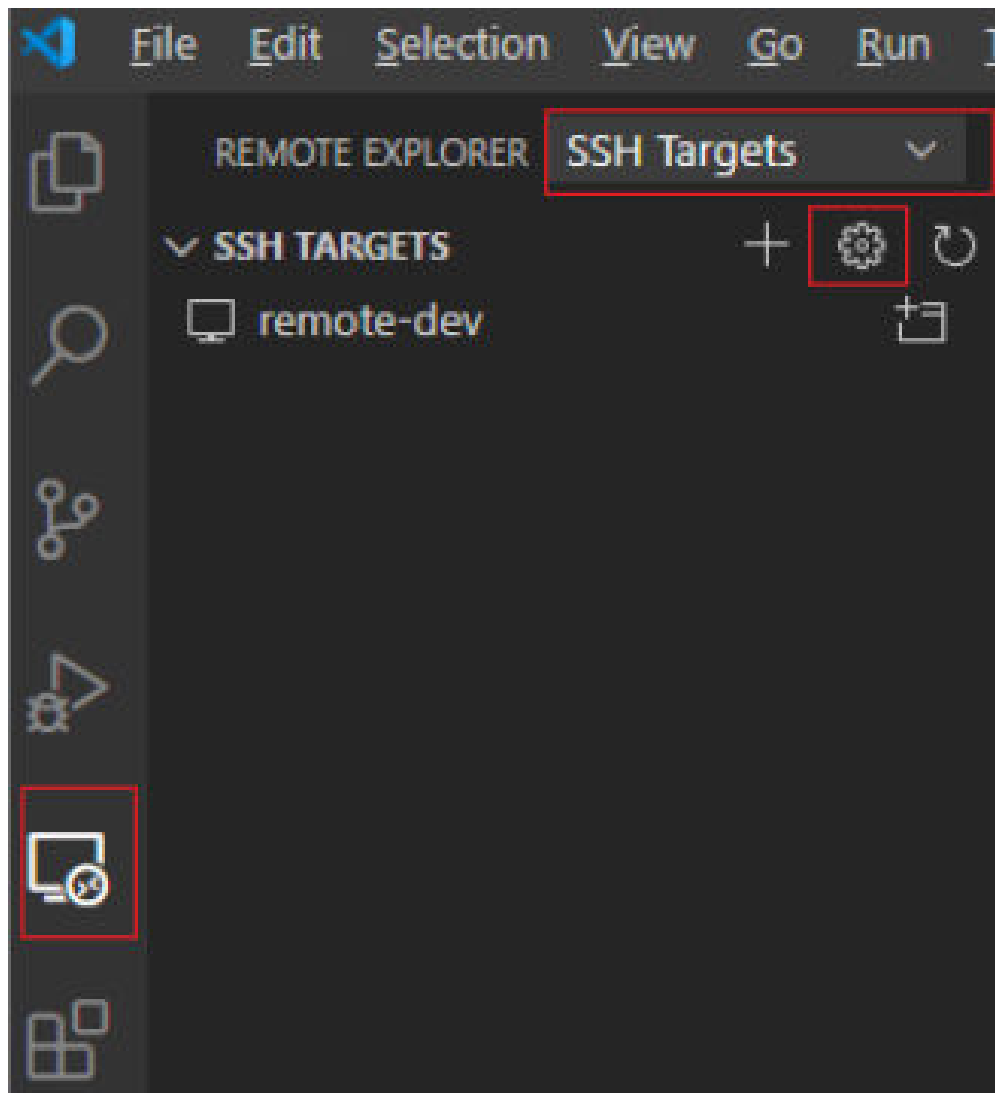
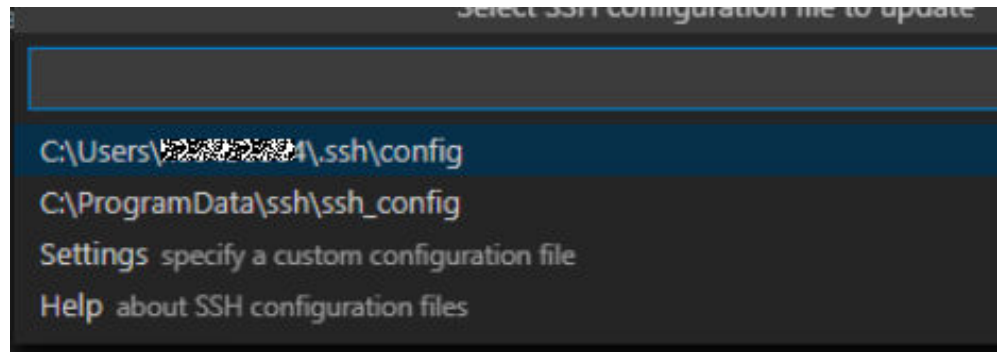
1. In the local VS Code development environment, click  on the left, select **SSH Targets** from the drop-down list box, and click . The SSH configuration file path is displayed.

Figure 5-50 Configuring SSH Targets



2. Click the SSH configuration path and configure SSH.

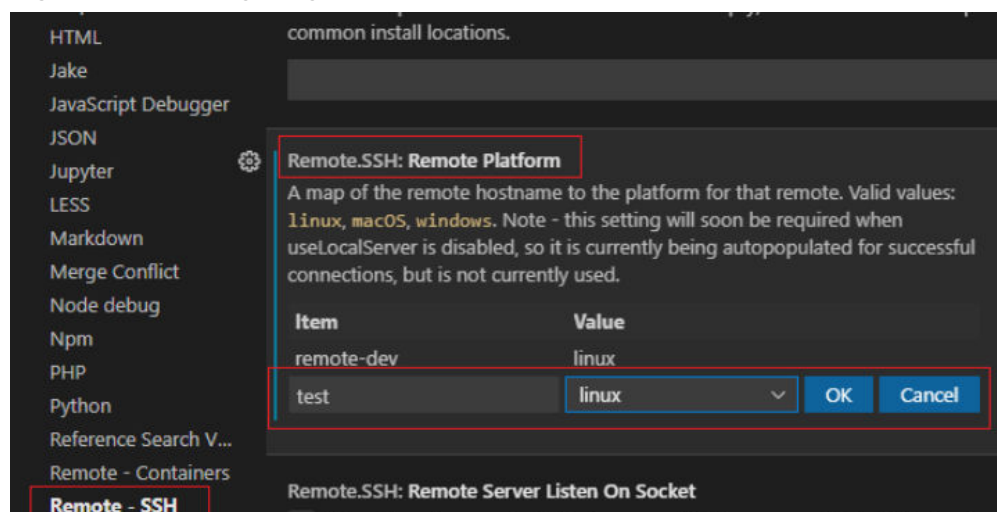
Figure 5-51 SSH configuration file path



```
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. Choose **File > Preference > Settings > Extensions > Remote-SSH**. On the **Remote Platform** page, click **Add Item**, set **Item** and **Value**, and click **OK**.

Figure 5-52 Configuring Remote Platform



Item: host name configured in SSH configuration
Value: remote development environment platform


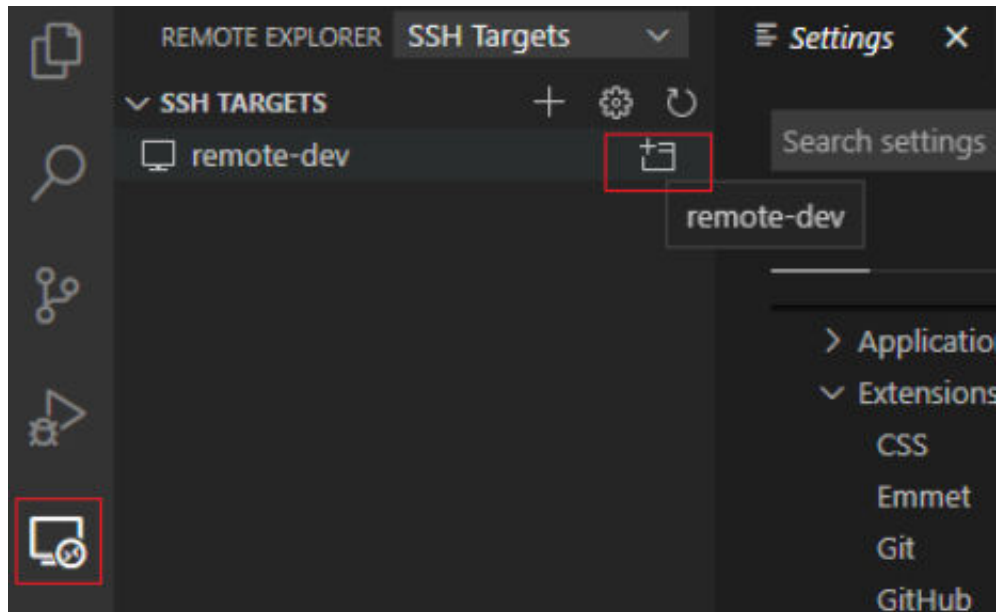
4. Go back to the **SSH Targets** page and click  on the right. Then, click the development environment name to open the development environment.

Figure 5-53 Opening the development environment



After the page shown in the following figure is displayed, the connection is succeeded.

Figure 5-54 Remote connection succeeded

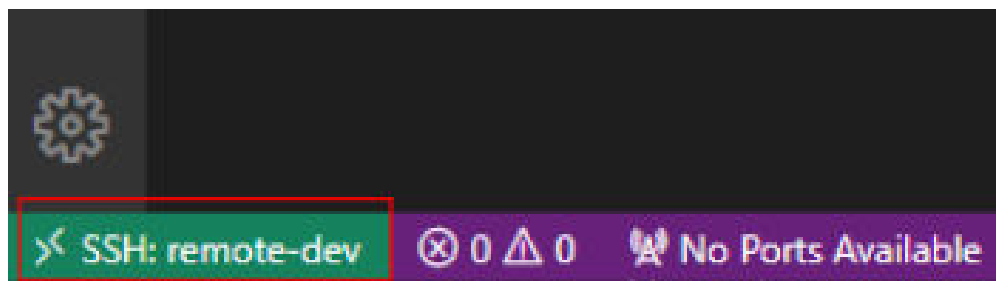
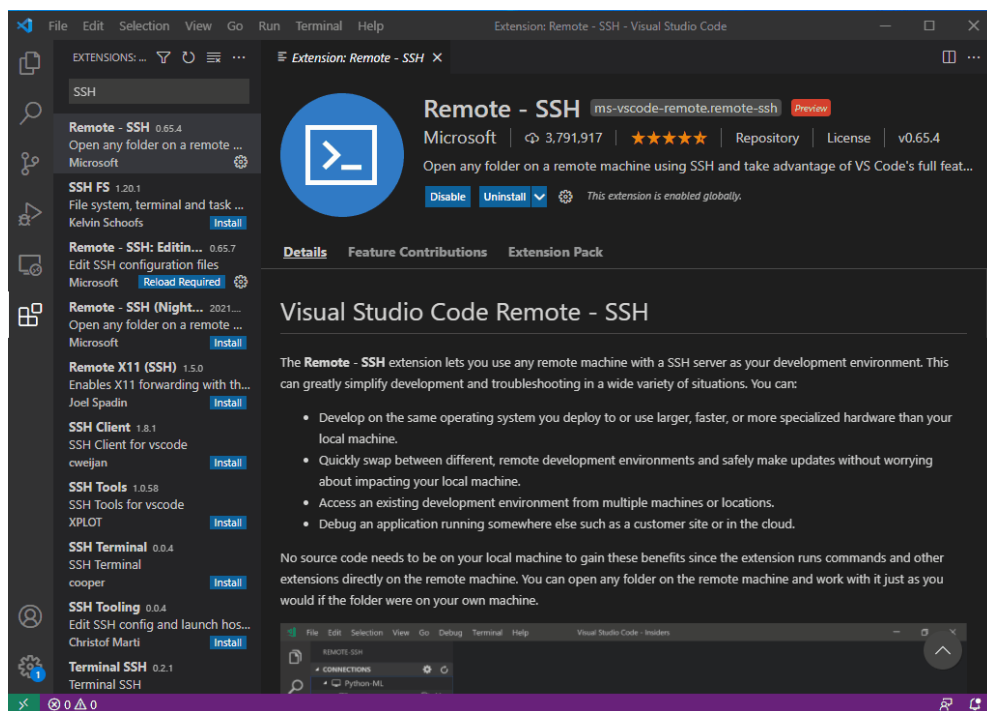


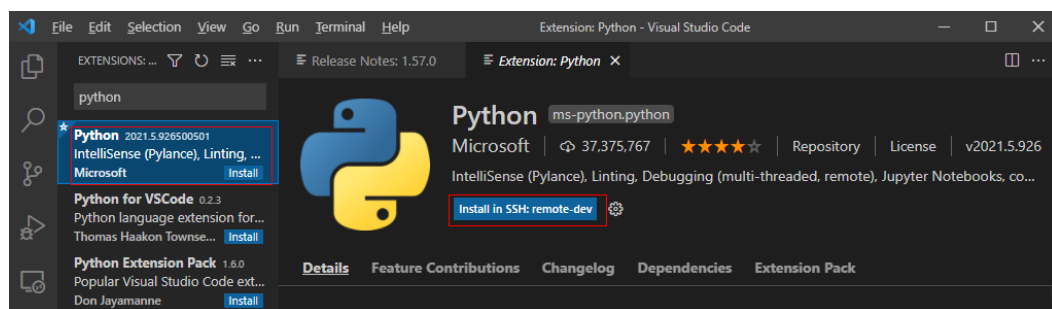
Figure 5-55 Complete configuration example



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 5-56 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
```

5.3.5 Remotely Debugging in VS Code

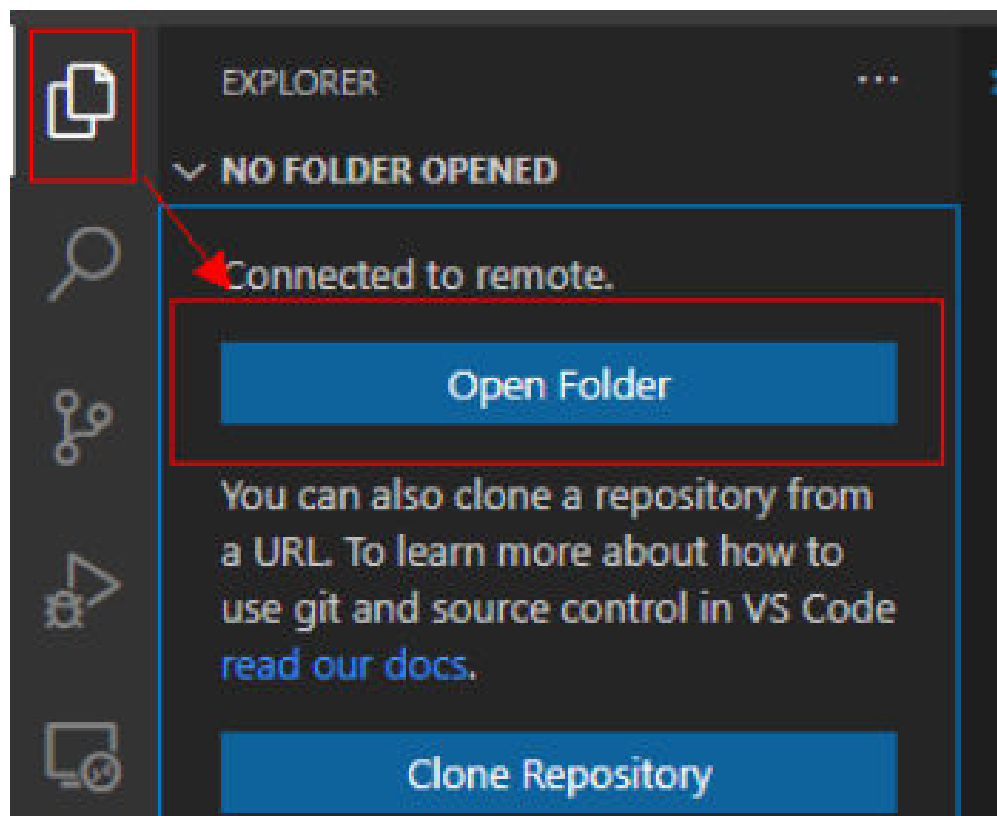
Prerequisites

A notebook instance has been accessed through VS Code.

Step 1 Upload Local Code to the Cloud Development Environment

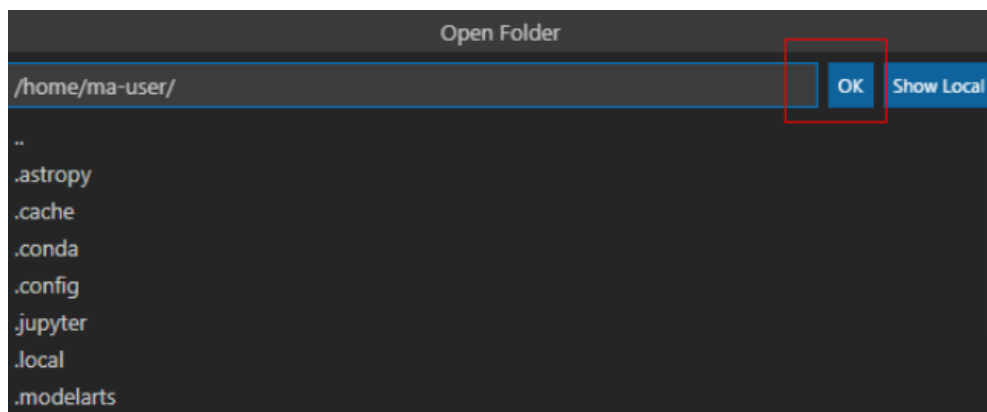
1. On the VS Code page, choose **File > Open Folder** to access the cloud path.

Figure 5-57 Open Folder



2. Select a path and click **OK**.

Figure 5-58 Selecting a file path

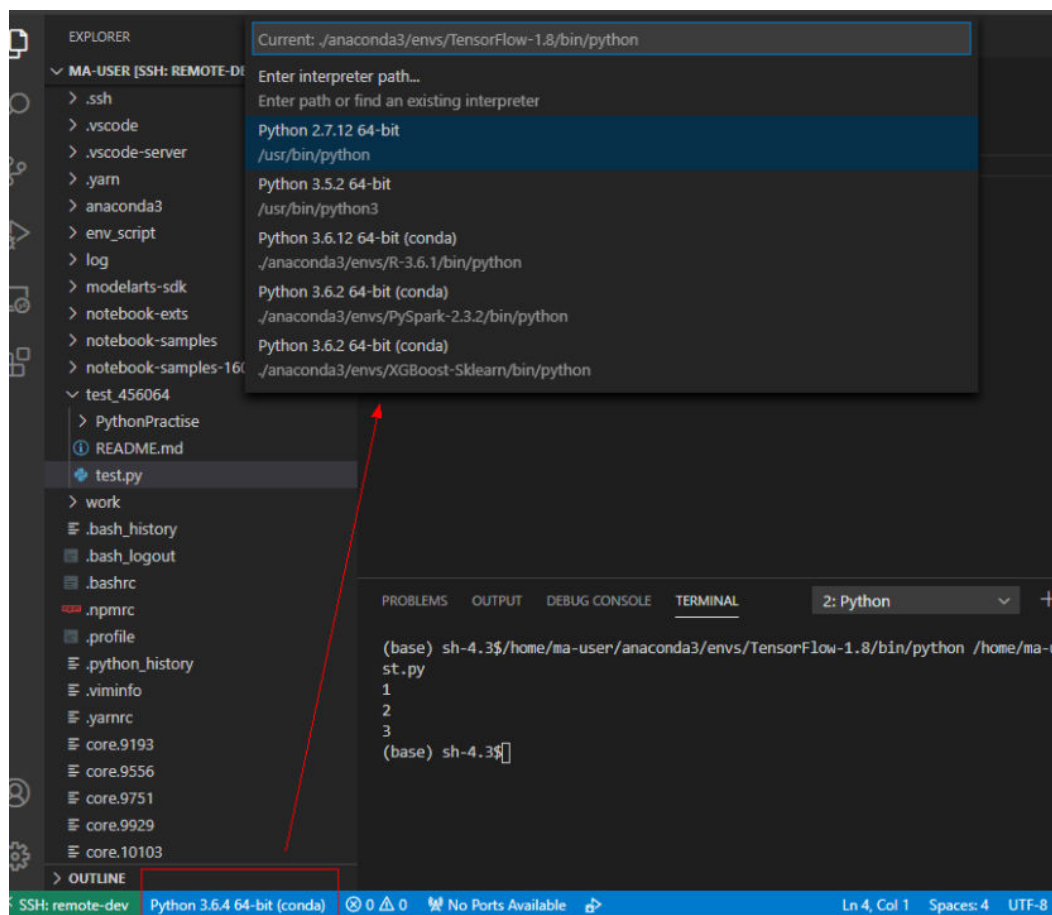


3. In the displayed directory structure on the left of the IDE, drag the code and files you want to upload to the corresponding folders. Then, the code is uploaded to the cloud development environment.

Step 2 Debug Code Remotely

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 5-59 Selecting a Python version



- Click the execution button to run the code. The code output is shown on the **TERMINAL** tab page.
- 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:
 - a. Choose **Run > Run and Debug** on the left.
 - b. Select the default Python code file.
 - c. Click on the left of the code to set breakpoints.
 - d. 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.

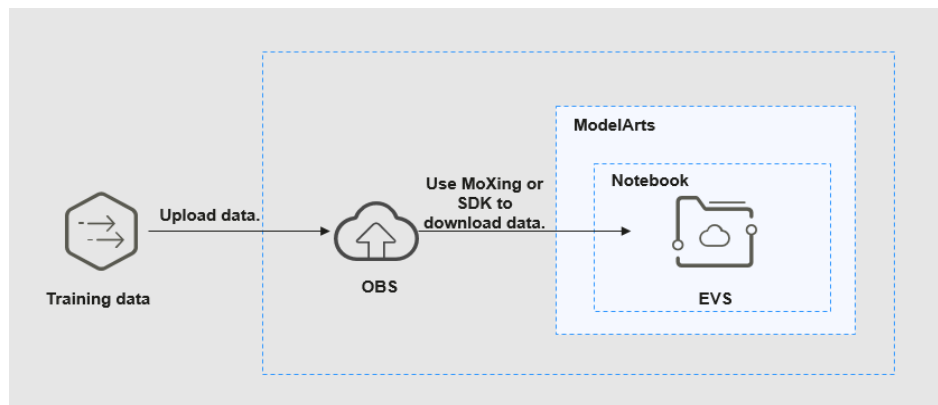
5.3.6 Uploading and Downloading Files in VS Code

Uploading Data from a Local IDE to a Notebook Instance

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.

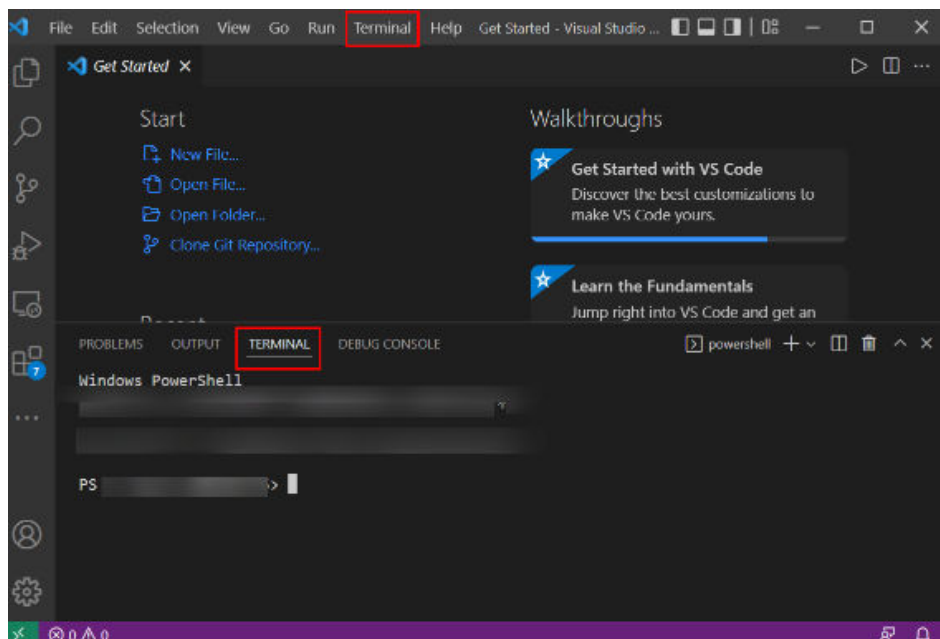
Figure 5-60 Uploading data to a notebook instance through OBS



Procedure

1. Upload data to OBS. For details, see [Uploading an Object](#). Alternatively, use ModelArts SDK on a local VS Code terminal.
Open the terminal in the local VS Code environment.

Figure 5-61 Opening the terminal in the local VS Code environment



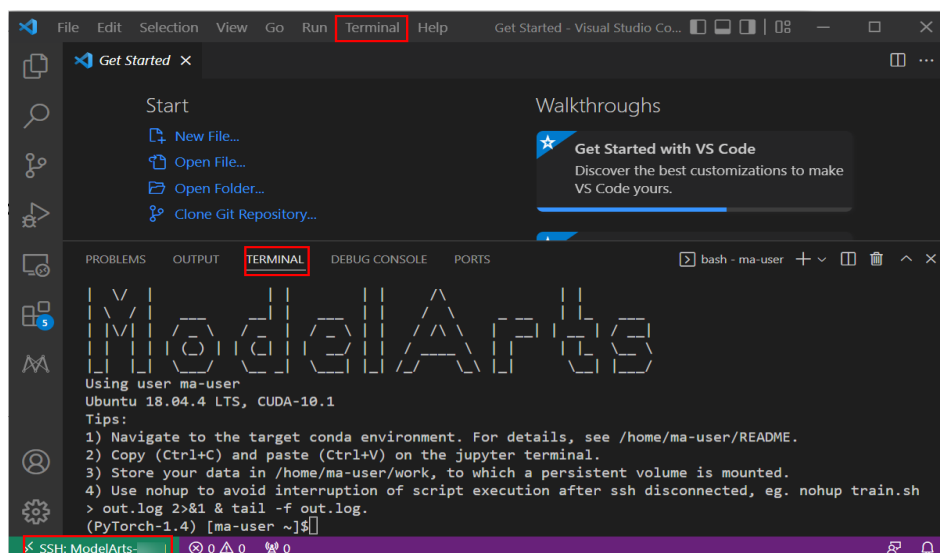
Enter **python** and press **Enter** to access the Python environment.

```
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. Use ModelArts SDK in the terminal of the remote VS Code environment to download the file from OBS to a development environment.

Figure 5-62 Opening the terminal in the remote VS Code environment



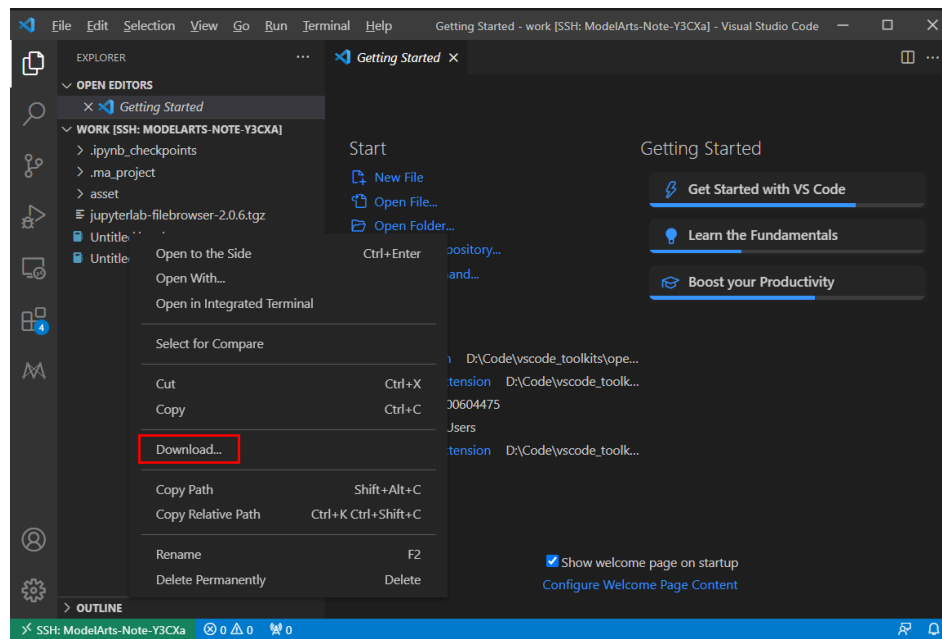
```
# Manually access the development environment using the source command.
cat /home/ma-user/README
# Select the target environment.
source /home/ma-user/miniconda3/bin/activate MindSpore-python3.7-aarch64
# Enter python and press Enter to access the Python environment.
python
```

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 5-63 Downloading files from a notebook instance to a local directory in VS Code



5.4 Local IDE (Accessed Using 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](#).
- The address and port number of the development environment are available. To obtain this information, go to the notebook instance details page.

Figure 5-64 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.

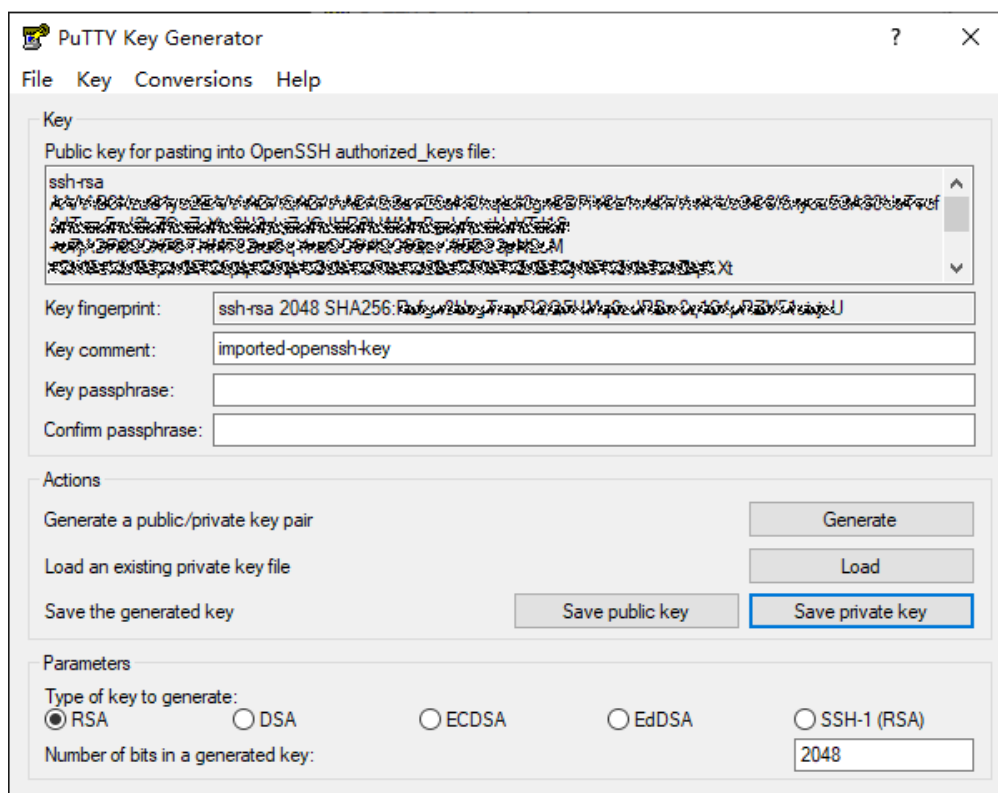
Step 1 Install the SSH Tool

[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 5-65 Converting the .pem key pair file to a .ppk key pair file

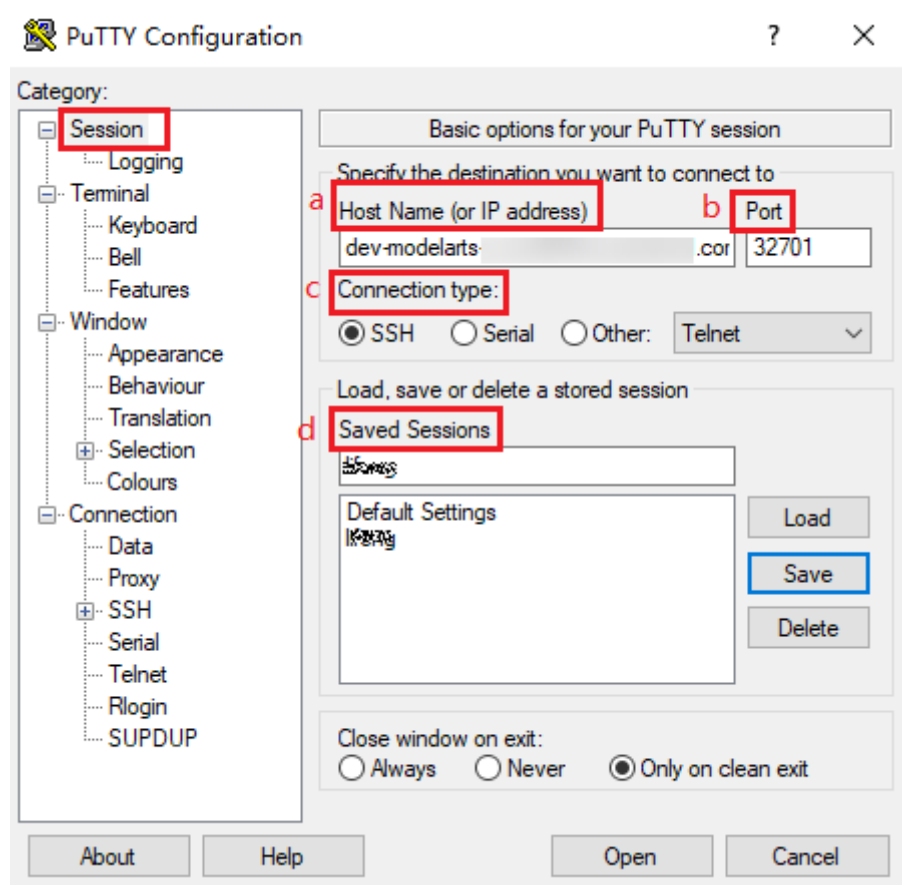


Step 3 Use SSH to Connect to a Notebook Instance

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 .

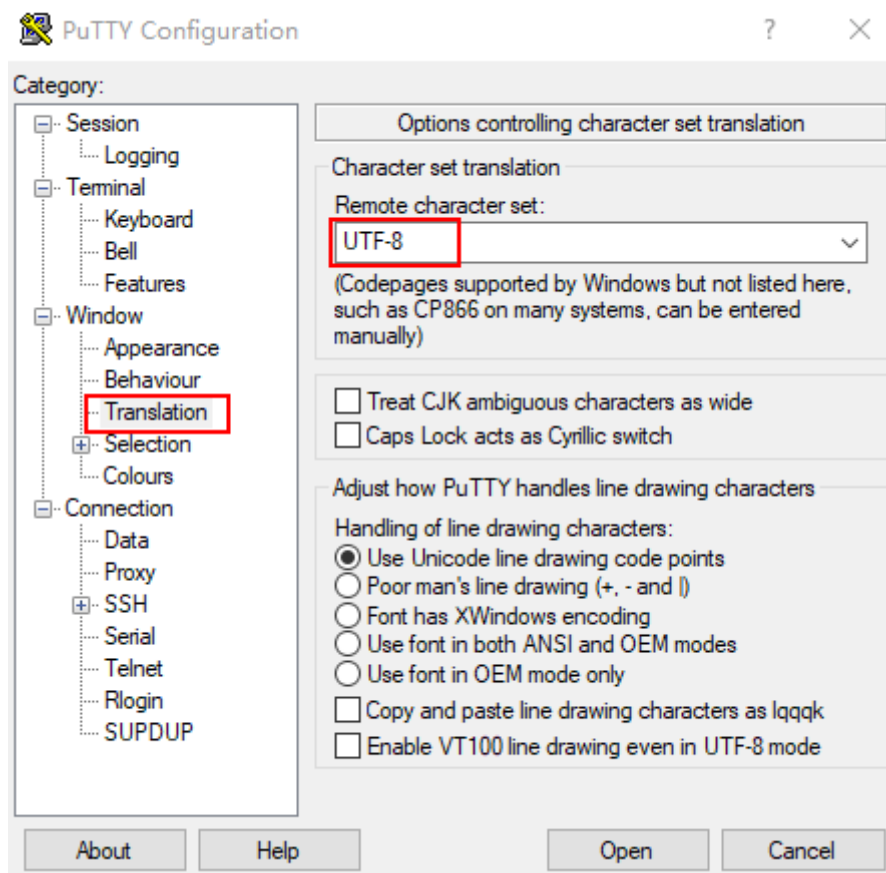
- b. **Port:** port number for accessing the in-cloud notebook instance. Obtain the port number on the page providing detailed information of the target notebook instance, for example, **32701**.
- c. **Connection type: SSH**
- d. **Saved Sessions:** task name, which can be clicked for remote access when you use PuTTY next time

Figure 5-66 Configuring Session



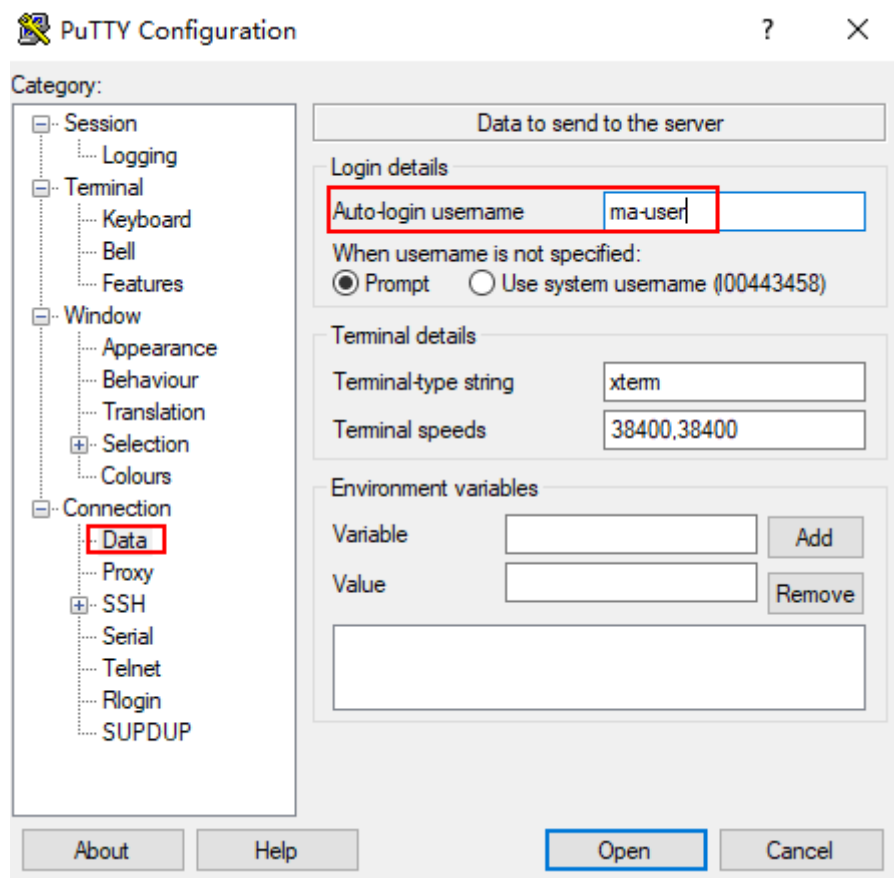
- 3. Choose **Window > Translation** and select **UTF-8** from the drop-down list box in the **Remote character set** area.

Figure 5-67 Setting the character format

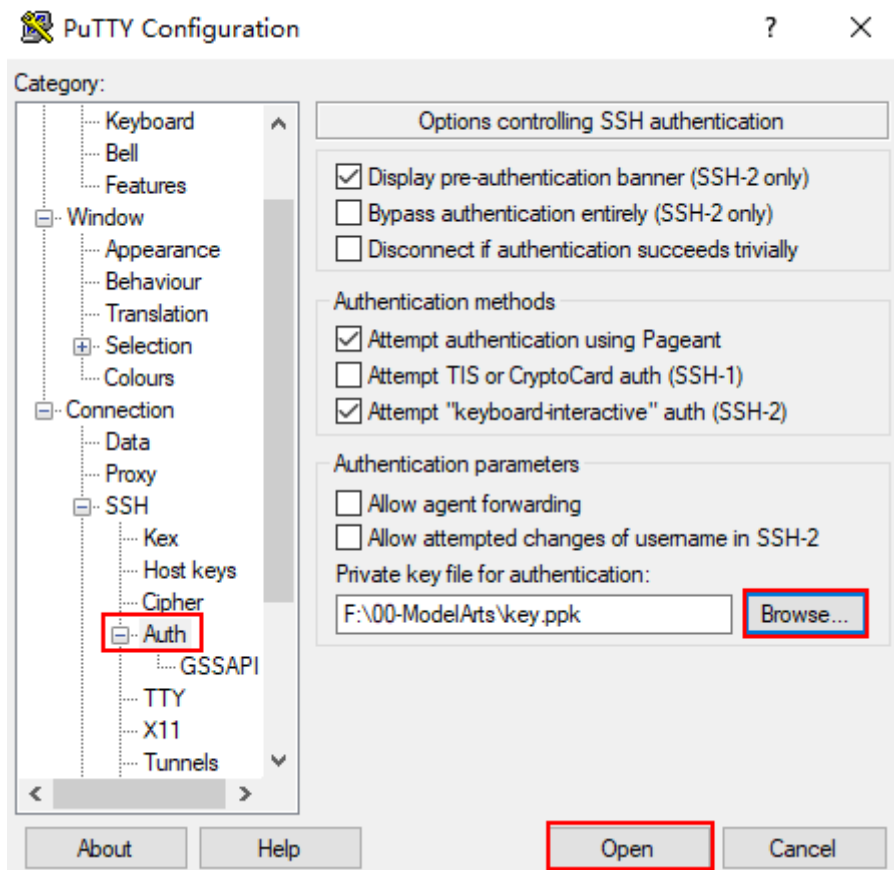


4. Choose **Connection > Data** and enter **ma-user** for **Auto-login username**.

Figure 5-68 Entering a username

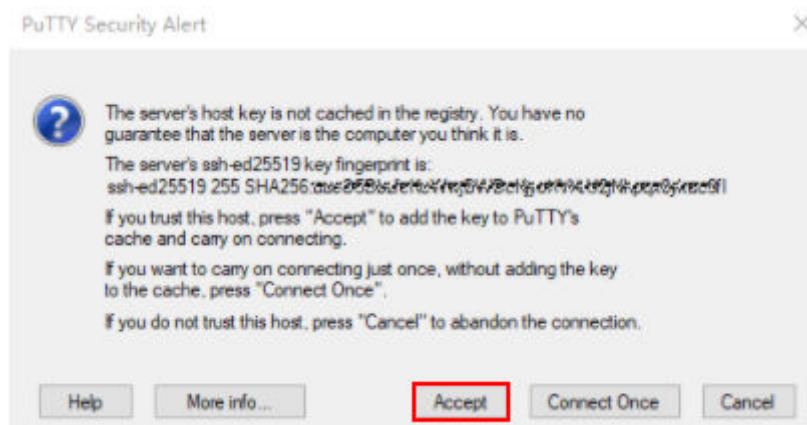


5. Choose **Connection** > **SSH** > **Auth**, click **Browse**, and select the .ppk file generated in [step 2](#).



6. 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 5-69 Asking if you want to accept the instance security certificate



7. Connect to the notebook instance.

Figure 5-70 Connecting to a notebook instance

```
Using username "ma-user".
Authenticating with public key "imported-openssh-key"
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 3.10.0-862.14.1.5.h328.eulerosv2r7.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$
```


6 ModelArts CLI Command Reference

[ModelArts CLI Overview](#)

[\(Optional\) Installing ma-cli Locally](#)

[Autocompletion for ma-cli Commands](#)

[ma-cli Authentication](#)

[ma-cli Image Building Command](#)

[Using the ma-cli ma-job Command to Submit a ModelArts Training Job](#)

[Using the ma-cli dli-job Command to Submit a DLI Spark Job](#)

[Using ma-cli to Copy OBS Data](#)

6.1 ModelArts CLI Overview

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 **DevEnviron** > **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, -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 6-1 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

Command	Description
dli-job	DLI Spark job submission and resource management
auto-completion	Command autocomplete

6.2 (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.

```
$openssl cms -verify -binary -in package.tar.gz.cms -signer "root" -inform DER -content package.tar.gz -noverify > ./test
st
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 *.*.*
```
2. Run **pip --version** to check whether the general package management tool pip is available.

```
C:\Users\xxx>pip --version
pip *.*.* from c:\users\xxx\appdata\local\programs\python\python*\lib\site-packages\pip (python *.*.*)
```
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.*.*
```

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.

6.3 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)
```

6.4 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 6-2 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

Parameter	Type	Man dator y	Description
-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
-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.

6.5 ma-cli Image Building Command

6.5.1 ma-cli Image Building Command

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 6-3 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.
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.)

6.5.2 Obtaining an Image Creation Template

ma-cli provides some common image creation templates, in which the guidance for developing Dockerfiles on ModelArts notebook is provided.

```
$ 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 6-4 Parameters

Parameter	Type	Mandatory	Description
--filter	String	No	Filter templates based on the template name keyword.
-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 .

Examples

Obtain an image creation 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
```

6.5.3 Loading an Image Creation 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 run the **--dest** command to specify the path. If a template folder with the same name already exists in the target path, run 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 6-5 Parameters

Parameter	Type	Mandatory	Description
--dst	String	No	Load templates to a specified path. 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.

Examples

Load the `customize_from_ubuntu_18.04_to_modelarts` image creation 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 ]
```

6.5.4 Obtaining 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 and 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 6-6 Parameters

Parameter	Type	Man dato ry	Description
-t / --type	String	No	Type of the images to be obtained. The options are BUILD_IN , DEDICATED , and ALL . <ul style="list-style-type: none"> • BUILD_IN: preset images • DEDICATED: custom images registered with ModelArts • ALL: all images
-f / --filter	String	No	Keyword of an image name, which is used to filter images
-v / --verbose	Bool	No	Whether to display detailed information. This function is disabled by default.
-i / --image-id	String	No	Obtain details about an image with a specified ID.
-n / --image-name	String	No	Obtain details about an image with a specified name.
-wi / --workspace-id	String	No	Obtain images in a specified workspace.
-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 .

Examples

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	fc5e3d002f 0314test	huaweicloud.com/notebook_test/0314test:1.0.0
2	193b2557	d39093a811 0328	7.myhuaweicloud.com/notebook_test/0328:1
3	171fe036	3b37e9aa7c 0926	aweicloud.com/ei_modelarts_y00218826_05/0926:1
4	1b48bb0a	689b0a7267 0926 swr	weicloud.com/ei_modelarts_y00218826_05/0926:111
5	c8667cf0	d2e3563107 1	huaweicloud.com/ei_modelarts_y00218826_05/1:6
6	3e6cda6a	a360eea80e 1	huaweicloud.com/ei_modelarts_y00218826_05/1:1
7	42e86ca5	ec198be968 111	.myhuaweicloud.com/notebook_test/111:1227
8	0f349cef	c411011ef2 11111110801	aweicloud.com/notebook_test/11111110801:111111
9	3a082e32	4f485aad6b 112121 swr	eicloud.com/ei_modelarts_y00218826_05/112121:123
10	db002f6	74eb00e1ce 1203	myhuaweicloud.com/notebook_test/1203:1.2.3
11	031dc02e	fd92cd457d8 1227	.myhuaweicloud.com/notebook_test/1227:111
12	f7d95648	7aaec8b1cc 1227	.myhuaweicloud.com/notebook_test/1227:888
13	2f720610	a1d1db9d7d 1227	.myhuaweicloud.com/notebook_test/1227:6666
14	42221bf2	22d726d270 1229	.myhuaweicloud.com/notebook_test/1229:123
15	70deea1e	70b2414ae7 123	myhuaweicloud.com/mindspore-dis-train/123:2
16	e6cc5414	ce318069f4 123	.myhuaweicloud.com/notebook_test/123:45678
17	6e7a86c9	319fb3bb28 1234	.myhuaweicloud.com/notebook_test/1234:666
18	ec036306	8c9dc6b391 1234	7.myhuaweicloud.com/notebook_test/1234:1
19	b7f8f3b	7a9941c978 441211	.myhuaweicloud.com/notebook_test/441211:11
20	d5acd51b	1ef16534d68 aaa	.myhuaweicloud.com/notebook_test/aaa:1.1.1

6.5.5 Creating an Image in ModelArts Notebook

Run the **ma-cli image build** command to create an image based on a specified Dockerfile. This command is available only in 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 6-7 Parameters

Parameter	Type	Mandatory	Description
FILE_PATH	String	Yes	Directory 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 creating an image.
--context	String	No	Path of the context information for data copying when creating a Dockerfile
-arg / --build-arg	String	No	Parameter for creating an image. If there are multiple parameters, run --build-arg VERSION=18.04 --build-arg ARCH=X86_64 .
-obs / --obs-path	String	No	Automatically upload the generated TAR package to OBS.
-f / --force	Bool	No	Whether to forcibly overwrite an existing SWR image with the same name. By default, the SWR image is not overwritten.

Examples

Create 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
-> [internal] load build definition from Dockerfile
-> [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
-> resolve swr.cn-north-7.myhuaweicloud.com/atelier/ubuntu:18.04
sha256:2918811864227c2f42aa9a3d5f53b1d469f67e2cf7e601f631b11961f7f8478 / 847B / 847B
sha256:bc38eaf959414179228aa4f42889396e4af24a9566c018311e0b355e353789 / 35.37kB
sha256:3659526d6cc64eeb1010bd2112e6f73981e1a8246e4f64e287763b57f101bb / 161B / 161B
sha256:23884877105a7f84a910895c4844051a551385ff6c36488ee880b76ec0e771 / 26.69MB / 26.69MB
-> extracting sha256:23884877105a7f84a910895c4844051a551385ff6c36488ee880b76ec0e771
-> extracting sha256:bc38eaf959414179228aa4f42889396e4af24a9566c018311e0b355e353789
-> extracting sha256:2918811864227c2f42aa9a3d5f53b1d469f67e2cf7e601f631b11961f7f8478
-> extracting sha256:3659526d6cc64eeb1010bd2112e6f73981e1a8246e4f64e287763b57f101bb
[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:b239078457d7c75d57a45989cf8d9d8e6fd9dc82a4ede6d4311bc487d8be9
-> exporting config sha256:6794fa8aebcc9464bf1102345237559fbd2a3774963098954841b1340cd51db
-> pushing layers
-> pushing manifest for swr.cn-north-7.myhuaweicloud.com/notebook_test/my_image:0.0.1
[auth] notebook_test/my_image:pull,push token for swr.cn-north-7.myhuaweicloud.com
*****
Summary Board
* Image Build Time: 4.3s
* Repository: swr.cn-north-7.myhuaweicloud.com/notebook_test/my_image
* Tags: 0.0.1
* Compressed Image Size: 25MB
* SWR Download Command: docker pull swr.cn-north-7.myhuaweicloud.com/notebook_test/my_image:0.0.1
(PyTorch-1.8) [ma-user work]$
```

6.5.6 Obtaining Image Creation Caches in ModelArts Notebook

Run the **ma-cli image df** command to obtain image creation caches. This command is available only in 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 6-8 Parameters

Parameter	Type	Mandatory	Description
-v / --verbose	Bool	No	Whether to display detailed information. This function is disabled by default.

Examples

- 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
iwrwrws19pdcjafe1ij6d0r918      true         98.50MB
cp52c4q81ud2abu2vp7sj5vyt       true         1.04MB
4jbo6v06r2w1575ddq3w8g12e       true         139.68kB
ojdjw5mok71s1nh2cauant051       true         86.86kB
k2jm6g061n5twmz7gmonmqjsh       true         16.55kB
efu5kwgig1ve44fe7smbrncnh*      true         8.19kB
uzikwqk5taxns1vajm14jrbje*      true         4.10kB
2g8p0qcb014g3qva7ucawkv87*     true         4.10kB
Reclaimable: 99.80MB
Total: 99.80MB
```

- View details about an image.

```
ma-cli image df --verbose
```

```
(PyTorch-1.8) [ma-user work]$ma-cli image df --verbose
ID: iwrns19pdcjafel1j6d0r918
Created at: 2023-03-28 12:23:28.353759532 +0000 UTC
Mutable: false
Reclaimable: true
Shared: false
Size: 98.50MB
Description: pulled from swr .....myhuaweicloud.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: cp52c4q81ud2abu2vp7s35vyt
Parents: iwrns19pdcjafel1j6d0r918
Created at: 2023-03-28 12:23:28.366910223 +0000 UTC
Mutable: false
Reclaimable: true
Shared: false
Size: 1.04MB
Description: pulled from swr .....myhuaweicloud.com/atelier/ubuntu:18.04@sha256:b5874.....2e235eea65d84f
Usage count: 1
Last used: 2023-03-28 12:23:28.38560437 +0000 UTC
Type: regular

ID: 4jbo6v06r9w1575ddq3uqg12e
Parents: k2jndq861nstwm7gnoomqjsh
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
```

6.5.7 Clearing Image Creation Caches in ModelArts Notebook

Run the **ma-cli image prune** command to clear image creation caches. This command is available only in 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 6-9 Parameters

Parameter	Type	Mandatory	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	Whether to retain the latest caches and clear only historical caches. The unit can be s (second), m (minute), or h (hour). The default value is 0, indicating that all caches will be cleared.
-v / --verbose	Bool	No	Whether to display detailed information. This function is disabled by default.

Examples

Retain 1 MB of image cache when clearing caches.

```
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
ojdjw5mok71s1nh2cauant05l      true         86.86kB
cp52c4q81ud2abu2vp7sj5vyt      true         1.04MB
iwrwwsi9pdcjafeli6d0r918       true         98.50MB
Total: 99.79MB
```

6.5.8 Registering SWR Images 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 the image 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 6-10 Parameters

Parameter	Type	Mandatory	Description
-swr / --swr-path	String	Yes	SWR path to the image to be registered

Parameter	Type	Mandatory	Description
-a / --arch	String	No	Architecture of the registered image. The value can be X86_64 or AARCH64 . The default value is X86_64 .
-s / --service	String	No	Service type of the registered image. The value can be NOTEBOOK or MODELBOX . The default value is NOTEBOOK . You can also specify both values, -s NOTEBOOK -s MODELBOX .
-rs / --resource-category	String	No	Resource type that can be used by the registered image. The value can be CPU , GPU , or ASCEND . The default value is CPU and GPU .
-wi / --workspace-id	String	No	Register an image into a specified workspace. The default workspace ID is 0 .
-v / --visibility	Bool	No	Available scope of the registered image. The value can be PRIVATE (available only to the image owner) or PUBLIC (available to all users). The default value is PRIVATE .
-td / --to-dli	Bool	No	Register an image with DLI.
-d / --description	String	No	Describe an image. By default, this parameter is left blank.

Examples

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-np-myhuaweicloud.com/notebook-0.0.1 /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": 26735897,
  "status": "ACTIVE",
  "swr_path": "swr.cn-np-myhuaweicloud.com/notebook-0.0.1",
  "tag": "0.0.1",
  "tags": [],
  "type": "DEDICATED",
  "update_at": "1680006812157",
  "visibility": "PRIVATE",
  "workspace_id": "0"
}
```

6.5.9 Deregistering a Registered Image from ModelArts Image Management

Run the **ma-cli image unregister** command to deregister a registered 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 6-11 Parameters

Parameter	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. This function is disabled by default.

Examples

Deregister a registered image from ModelArts image management.

```
ma-cli image unregister --image-id=xx
```

```
(PyTorch-1.8) [ma-user work]$ma-cli image unregister --image-id=852f85c1590a66748
[ OK ] Successfully unregistered image 852f85dd-acd590a66748
```

6.5.10 Debugging an SWR Image on an ECS

ma-cli allows you to debug an SWR image on an ECS to determine whether to use the image in a ModelArts development environment.

```

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 6-12 Parameters

Parameter	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 debugged image. The value can be NOTEBOOK or MODELBOX . The default value is NOTEBOOK .
-a / --arch	String	No	Architecture of the debugged image. The value can be X86_64 or AARCH64 . The default value is X86_64 .
-g / --gpu	Bool	No	GPU debugging status. This function is disabled by default.

6.6 Using the ma-cli ma-job Command to Submit a ModelArts Training Job

6.6.1 ma-cli ma-job Command Overview

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 jod 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 6-13 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.

6.6.2 Obtaining ModelArts Training Jobs

Run the **ma-cli ma-job get-job** command to view 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 6-14 Description

Parameter	Type	Mandatory	Description
-i / --job-id	String	No	Obtain details about a training job with a specified job ID.
-n / --job-name	String	No	Obtain a training job with a specified job name or filter training jobs by job name.
-pn / --page-num	Int	No	Page number. The default value is page 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. This function is disabled by default.

Examples

- Obtain a training task job 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_ed3a963f-5438-4a99-9a19-c97ce88c48bb | Comple | ei_modela | 00h:01m:16s | 2023-03-29 03:41:21 | 2023-03-29 03:41:36 | |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

- Filter training jobs by job name **auto**.

```
ma-cli ma-job get-job -n auto
```

```
(PyTorch-1.4) [ma-user work]$ma-cli ma-job get-job -n auto
```

index	id	name	status	user_name	duration	create_time	start_time
1	9b495c-	autotest_oh278-copy-4582	Completed	ei_modelarts_y0021882_6_05	00h:01m:31s	2023-03-29 07:03:08	2023-03-29 07:05:20
2	af2147f5-	autotest_oh278-copy-ae52	Terminated	ei_modelarts_y0021882_6_05	00h:10m:49s	2023-03-29 06:52:16	2023-03-29 06:52:32
3	2c1855b1-	autotest_nv487q	Failed	ei_modelarts_y0021882_6_05	00h:37m:29s	2023-03-29 03:22:31	2023-03-29 03:22:58
4	4525b3c9-	autotest_x2cjf6	Failed	ei_modelarts_y0021882_6_05	00h:00m:01s	2023-03-29 03:19:41	2023-03-29 03:19:49
5	4234455d-	autotest_sx71zc	Terminated	ei_modelarts_y0021882_6_05	00h:00m:00s	2023-03-29 02:25:18	N/A
6	9810ae49-	autotest_s62gz3	Terminated	ei_modelarts_y0021882_6_05	00h:00m:06s	2023-03-29 02:19:49	2023-03-29 02:20:13
7	90c7de89-	autotest_wf8z2g	Abnormal	ei_modelarts_y0021882_6_05	00h:00m:00s	2023-03-29 01:43:18	N/A
8	fc740dc5-	autotest_g17mit	Terminated	ei_modelarts_y0021882_6_05	00h:00m:00s	2023-03-29 01:22:19	N/A
9	5d16fdfe-	autotest_02dfd461	Terminated	ei_modelarts_y0021882_6_05	00h:00m:00s	2023-03-29 01:11:26	N/A
10	3737e56d-	autotest_clutp0	Completed	ei_modelarts_y0021882_6_05	00h:05m:59s	2023-03-29 00:59:28	2023-03-29 01:04:20

6.6.3 Submitting a ModelArts Training Job

Run the **ma-cli ma-job submit** command to submit a ModelArts training job.

Before running this command, configure **YAML_FILE** 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 the **option** parameter of the command. If you specify both the **YAML_FILE** configuration file and the **option** parameter in the CLI, the value of the **option** parameter will overwrite that 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 6-15 Parameters

Parameter	Type	Mandatory	Description
YAML_FILE	String	No	Configuration file of a training job. If this parameter is not specified, the configuration file is empty.
--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 is used to submit a training job. This parameter can be omitted when you use a custom image or 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	Runtime UID of a custom image. The default value is 1000 .
--working-dir	String	No	Work directory where an algorithm is executed

Parameter	Type	Mandatory	Description
<code>--local-code-dir</code>	String	No	Local directory to the training container to which the algorithm code directory is downloaded
<code>--user-command</code>	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.
<code>--pool-id</code>	String	No	Resource pool ID selected for a training job. To obtain the ID, do as follows: Log in to the ModelArts management console, choose Dedicated Resource Pools in the navigation pane on the left, and view the resource pool ID in the dedicated resource pool list.
<code>--train-instance-type</code>	String	No	Resource flavor selected for a training job
<code>--output</code>	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 .
<code>--input</code>	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 .
<code>--env-variables</code>	String	No	Environment variables input during training. To specify multiple parameters, use --env-variables ENV1=env1 --env-variables ENV2=env2 .
<code>--parameters</code>	String	No	Training input parameters. To specify multiple parameters, use --parameters "--epoch 0 --pretrained" .
<code>--framework-type</code>	String	No	Engine selected for a training job

Parameter	Type	Mandatory	Description
--framework-version	String	No	Engine version selected for a training job
-q / --quiet	Bool	No	After a training job is submitted, the system exits directly and does not print the job status synchronously.
--workspace-id	String	No	Workspace where a training job is deployed. The default value is 0 .
--policy	String	No	Training resource specification mode. The options are regular , economic , turbo , and auto .
--volumes	String	No	Mount EFS disks. 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:/"

Submitting a Training Job Based on a Preset ModelArts Image

Submit a training job by specifying the **options** parameter in the CLI.

```
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
```

The following is an 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:/
```

Using a Custom Image to Create a Training Job

Submit a training job by specifying the **options** parameter in the CLI.

```
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
```

The following is an 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:/

```

Examples

- Submit a training job based on a YAML file.

```
ma-cli ma-job submit ./train-job.yaml
```

```
(PyTorch-1.4) [ma-user work]$ma-cli ma-job submit ./train_job.yaml
[ OK ] Current training job id is: 4d7c8584-b213-4f88-9833-d3e6a82f9e42
[ OK ] Creating
[ OK ] Running
```

- Submit a training job using preset image **pytorch1.8-cuda10.2-cudnn7-ubuntu18.04** through the CLI.

```
ma-cli ma-job submit --code-dir obs://automation-use-only/Original/TrainJob/TrainJob-v2/
pytorch1.8.0_cuda10.2/code/ \
  --boot-file test-pytorch.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://automation-use-only/Original/TrainJob/TrainJob-v2/
pytorch1.8.0_cuda10.2/data/ \
  --log-url obs://automation-use-only/Original/TrainJob/TrainJob-v2/
pytorch1.8.0_cuda10.2/data/logs/ \
  --train-instance-type modelarts.vm.cpu.8u \
  --train-instance-count 1 \
```

```
(PyTorch-1.4) [ma-user work]$ma-cli ma-job submit --code-dir obs://au
/Original/TrainJob/T
?/pytorch1.8.0_cuda10.2/code/ \
>
--boot-file test-pytorch.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://autome
y/Original/TrainJob/TrainJob-v2/pytorch1.8.0_cuda10.2/data/ \
>
--log-url obs://autome
/Original/TrainJob/TrainJob-v2/pytorch1.8.0_cuda10.2/data/logs/ \
>
--train-instance-type modelarts.vm.cpu.8u \
>
--train-instance-count 1 \
>
[ OK ] Current training job id is: 7db3e6f9-181d-4142-ba13-235213499430
[ OK ] Creating
[ OK ] Running
```

6.6.4 Obtaining 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	Type	Mandatory	Description
-i / --job-id	String	Yes	Obtain logs of a training job with a specified job ID.
-t / --task-id	String	No	Obtain logs of a specified task, which defaults to work-0 .

Examples

Obtain logs of a training job with a specified job ID.

```
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 event" file="report.go:10" 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:10" Command=bootstrap/init Component=ma-training-toolkit Platform=ModelArts-Service
time="2023-03-29T11:41:27+08:00" level=info msg="scc is already running" file="scc.go:10" 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:10" Command=bootstrap/init Component=ma-training-toolkit Platform=ModelArts-Service
time="2023-03-29T11:41:27+08:00" level=info msg="[init] running" file="init.go:10" Command=bootstrap/init Component=ma-training-toolkit Platform=ModelArts-Service
time="2023-03-29T11:41:27+08:00" level=info msg="[init] ip or" file="init.go:10" Command=bootstrap/init Component=ma-training-toolkit Platform=ModelArts-Service
time="2023-03-29T11:41:27+08:00" level=info msg="local dir = s" file="init.go:10" 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:10" 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
```

6.6.5 Obtaining ModelArts Training Job Events

Run the **ma-cli ma-job get-event** command to view 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	Type	Mandatory	Description
-i / --job-id	String	Yes	Obtain events of a training job with a specified job ID.

Examples

Obtain events of a training job with a specified job ID.

```
ma-cli ma-job get-event --job-id b63e90baxxx
```

```
(PyTorch-1.4) [ma-user work]ma-cli ma-job get-event --job-id b63e90baxxx
-----
| STAT | INFO | TIME |
-----
| [ ] | Training job completed. | 2023-03-29T11:4 |
| [2m] | | 2:47:08:00 |
| [OK] | | |
| [0m] | [worker-0][time used: 0.136s] Upload training output(parameter name: output_url) finished. | 2023-03-29T11:4 |
| [2m] | | 2:42:08:00 |
| [OK] | | |
| [0m] | [worker-0] Training output(parameter name: output_url) uploading. | 2023-03-29T11:4 |
| [2m] | | 2:42:08:00 |
| [OK] | | |
| [0m] | [Job: modelarts-job-b63e90ba-5 ] ExecuteAction: Start to execute action CompleteJob | 2023-03-29T11:4 |
| [2m] | | 2:42:08:00 |
| [OK] | | |
| [0m] | [worker-0] Training finished. Exit code 0. | 2023-03-29T11:4 |
| [2m] | | 2:40:08:00 |
| [OK] | | |
| [0m] | [worker-0] training started. | 2023-03-29T11:4 |
| [2m] | | 1:38:08:00 |
```

6.6.6 Obtaining ModelArts AI Engines for Training

Run the **ma-cli ma-job get-engine** command to obtain ModelArts AI engines for training.

```
$ 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 6-16 Parameters

Parameter	Type	Mandatory	Description
-v / --verbose	Bool	No	Whether to display detailed information. This function is disabled by default.

Examples

View the AI engine of a training job.

```
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	

6.6.7 Obtaining ModelArts Resource Specifications for Training

Run the **ma-cli ma-job get-flavor** command to obtain ModelArts resource specifications for training.

```
$ 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 6-17 Parameters

Parameter	Type	Mandatory	Description
-t / --flavor-type	String	No	Resource flavor. If this parameter is not specified, all resource flavors are returned by default.
-v / --verbose	Bool	No	Whether to display detailed information. This function is disabled by default.

Examples

View the resource flavor and type of a training job.

```
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 |
+-----+-----+-----+-----+
```

6.6.8 Stopping 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 6-18 Parameters

Parameter	Type	Mandatory	Description
-i / --job-id	String	Yes	Training job ID
-y / --yes	Bool	No	Whether to forcibly stop a specified training job

Examples

Stop a running training job.

```
ma-cli ma-job stop --job-id efd3e2f8xxx
```

6.7 Using the ma-cli dli-job Command to Submit a DLI Spark Job

6.7.1 Overview

```
$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 6-19 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.

Command	Description
get-queue	Obtain DLI queues.
get-resource	Obtain DLI group resources.
stop	Stop a DLI Spark job.
submit	Submit a DLI Spark job.
upload	Upload local files or OBS files to a DLI group.

6.7.2 Querying DLI Spark Jobs

Run **ma-cli dli-job get-job** to query the DLI Spark job list 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 6-20 Parameters

Parameter	Type	Mandatory	Description
-i / --job-id	String	No	ID of a DLI Spark job used to obtain job details

Parameter	Type	Mandatory	Description
-n / --job-name	String	No	Name of a DLI Spark job used to query the job or keywords contained in job names used to filter DLI Spark jobs
-pn / --page-num	Integer	No	Job index page. The default value is page 1.
-ps / --page-size	Integer	No	Number of jobs displayed on each page. The default value is 20 .
-v / --verbose	Bool	No	Whether to display detailed information. This function is disabled by default.

Example

Run the following command to query 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	12fbcc37-8df6	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	82856887-5bd3	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	d70717e2-1a36	zh	dead	dli_ma_notebook	CUSTOMIZED	notebook
11	85358931-99af	zh	dead	dli_ma_notebook	CUSTOMIZED	notebook
12	d5546f21-430e	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-cdb6	zh	dead	dli_ma_notebook	CUSTOMIZED	notebook
17	eff42ca1-074e	zh	dead	dli_ma_notebook	CUSTOMIZED	notebook
18	9357a261-72df	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

6.7.3 Submitting a DLI Spark Job

Run the **ma-cli dli-job submit** command to submit a DLI Spark job.

Before running this command, configure **YAML_FILE** 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 the **option** parameter of the command. If you specify both the **YAML_FILE** configuration file and the **option** parameter in the CLI, the value of the **option** parameter will overwrite that in the configuration file.

CLI Parameters

```
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 with **options** specified:

```
$ 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 6-21 Description

Parameter	Type	Mandatory	Description
YAML_FILE	String, a local file path	No	Configuration file of a DLI Spark job. If this parameter is not specified, the configuration file is empty.
--file	String	Yes	Entry file for program running. The value 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 job running.
-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 component version used by a job.

Parameter	Type	Mandatory	Description
-st / `--sc-type	String	No	If the current Spark component version is 2.3.2, leave this parameter blank. If the current Spark component 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 component 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 . <ul style="list-style-type: none"> • 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 6-23 .
-ec / --executor-cores	String	No	Number of CPU cores of each Executor in the Spark application. This configuration will replace the default setting in sc_type .
-em / --executor-memory	String	No	Executor memory of the 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 the 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	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	Input parameters of the main class, which are application parameters. To specify multiple parameters, use --args arg1 --args arg2 .
-q / --quiet	Bool	No	After a DLI Spark job is submitted, the system exits directly and does not print the job status synchronously.

Examples

- Submit a DLI Spark job using the **YAML_FILE** 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 in the CLI.

```
$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
```

6.7.4 Querying DLI Spark Run Logs

Run **ma-cli dli-job get-log** to query backend logs of DLI Spark jobs.

```
$ 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 6-22 Parameters

Parameter	Type	Mandatory	Description
-i / --job-id	String	Yes	ID of a DLI Spark job used to obtain job logs

Example

Run the following command to obtain run logs of a DLI Spark job using its job ID:


```
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          'bin/bash'
+ set -e
+ '[' -z o          n/bash ']'
+ SPARK CL
+ grep SPA
+ sort -t
+ sed 's/[
+ env
+ readanna
+ '[' -n '
+ '[' 3 ==
+ '[' 3 ==
++ python3
+ pyv3=Py
+ export P
+ PYTHON_V
+ export P
+ PYSARK
+ export P
+ PYSARK
+ '[' -z X
+ SPARK CL          pt/spark/jars/**
+ '[' -z '
+ case "$1
+ shift 1
+ CMD="(("$S          --conf "spark.driver.bindAddress=$SPARK_DRIVER_BIND_ADDRESS" --deploy-mode client "$@"
+ '[' true
+ '[' true
+ '[' -z X
+ '[' -z X
+ exec /us          in/spark-submit --conf spark.driver.bindAddress=172.30.0.62 --deploy-mode client --properties-
oy.PythonR          ib/submit_notebook.py
++ tee -a
++ tee -a
++ sed -u -e 's/\[[0-9;]*m//g' -e 's/\x1b//g'
```

6.7.5 Querying DLI Queues

Run `ma-cli dli-job get-queue` to query DLI queues.

```
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 6-23 Parameters

Parameter	Type	Mandatory	Description
-n / --queue-name	String	No	Name of a DLI queue to be queried
-t / --queue-type	String	No	Type of DLI queues to be queried. The value can be sql , general , or all . The default value is all .
-tags / --tags	String	No	Tags of DLI queues to be queried
-pn / --page-num	Integer	No	DLI queue page index. The default value is page 1.
-ps / --page-size	Integer	No	Number of DLI queues displayed on each page. The default value is 20 .

Example

Run the following command to query information about the **dli_ma_notebook** queue:

```
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': 'ma-user',
 'queueName': 'dli_ma_notebook',
 'queueType': 'general',
 'queue_id': 81111,
 'resource_id': '242e7af8-c9d1',
 'resource_mode': 1,
 'resource_type': 'container',
 'support_spark_versions': ['2.3.2', '2.4.5', '3.1.1']}
```

6.7.6 Obtaining 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 6-24 Parameters

Parameter	Type	Mandatory	Description
-n / --resource-name	String	No	Obtain details about DLI group resources by resource name.
-k / --kind	String	No	Obtain details about DLI group resources by resource type, which can be JAR, PyFile, file, or modelFile.
-g / --group	String	No	Obtain details about DLI group resources by group name.
-tags / --tags	String	No	Obtain details about DLI group resources by resource tag.

Examples

Obtain 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']}]}
```

6.7.7 Uploading Local Files or OBS 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 6-25 Parameters

Parameter	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 the file is to be uploaded
-tags / --tags	String	No	Tag of the file to be uploaded
-obs / --obs-bucket	String	No	If the file to be uploaded contains a local path, specify an OBS bucket for transit.
-async / --is-async	Bool	No	Asynchronously upload files. This method is recommended.

Examples

- 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.
```

6.7.8 Stopping 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.
-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 6-26 Parameters

Parameter	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

Examples

```
ma-cli dli-job stop -i ${your_job_id}
```

```
(PyTorch-1.8) [ma-user work]$ma-cli dli-job stop -i 4b38a0c5-861
[ WARN ] Spark job 4b38a0c5-861 will be stopped, do you want to continue (y for confirm)? [y/N]: y
[ OK ] Successfully stop spark batch job [ 4b38a0c5-861 ].
```

6.8 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 6-27 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/ ]
```