ModelArts

Best Practices

Issue	01
Date	2026-11-17



HUAWEI

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Copyright © Huawei Cloud Computing Technologies Co., Ltd. 2026. 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


and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

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

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

Huawei Cloud Computing Technologies Co., Ltd.

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

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

Contents

1 ModelArts Best Practices..... 1

2 LLM Inference..... 3

2.1 Deploying a Model from the vLLM-Ascend Open-Source Community on the ModelArts Inference Platform..... 3

3 LLM Training..... 20

3.1 Training the Qwen3 Series Models Adapted to the MindSpeed-LLM Framework on ModelArts..... 20
3.1.1 Overview..... 20
3.1.2 Configuring Cloud Resources..... 22
3.1.3 Preparing Model Weights and Training Data..... 23
3.1.4 Preparing the Training Scripts..... 24
3.1.5 Executing a Training Job..... 30
3.1.6 Viewing the Training Output..... 39
3.1.7 FAQs About Training..... 40
3.2 Training the Qwen3 & Qwen2.5-VL Model Adapted to the verl Framework on ModelArts..... 43
3.2.1 Overview..... 43
3.2.2 Configure Cloud Resources..... 45
3.2.3 Preparing Model Weights and Training Data..... 46
3.2.4 Preparing the Training Scripts..... 50
3.2.5 Executing a Training Job..... 61
3.2.6 Viewing the Training Output..... 66
3.3 Training the Qwen3 Series Models Adapted to the AReaL Framework on ModelArts..... 67
3.3.1 Overview..... 67
3.3.2 Configuring Cloud Resources..... 69
3.3.3 Preparing Model Weights and Training Data..... 70
3.3.4 Preparing the Training Scripts..... 71
3.3.5 Executing a Training Job..... 74
3.3.6 Viewing the Training Output..... 82
3.3.7 Debugging a Training Job..... 82

4 Video Generation Model Training and Inference..... 83

4.1 Inference Guide for Wan2.1, Wan2.2, HunyuanVideo, and CogVideo Series Models Adapted to NPU via ModelArts Lite Server (6.5.911)..... 83

5 Migrating Training and Inference from GPU to Ascend..... 100

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

5.1 GPU-to-NPU Migration: Handwritten Digit Recognition....100
6 Notebook....114
6.1 Migrating the Conda Environment on a Notebook Instance to an SFS Disk....114
7 Model Training....118
7.1 Running a GPU-based Training Job on ModelArts....118
7.1.1 Scenarios....118
7.1.2 Preparations....121
7.1.3 Running a Single-Node Single-PU Training Job on ModelArts....129
7.1.4 Running a Single-Node Multi-PU Training Job on ModelArts....144
7.1.5 Running a Multi-Node Multi-PU Training Job on ModelArts....154
8 Inference Deployment (New Version)....164
8.1 Accessing a Real-Time Service Through a Dedicated API Gateway, WAF, VPC, and ELB....164
8.2 Accessing a Real-Time Service Through ELB....182
9 Inference Deployment (Old Version)....190
9.1 Creating a Model Using a Custom Engine....190
9.2 Using a Large Model to Create a Model on ModelArts Standard and Deploy It as a Real-Time Service....193
9.3 Full-Process Development of WebSocket Real-Time Services....197
10 Permissions Management....203
10.1 Basic Concepts....203
10.2 Permission Management Mechanisms....208
10.2.1 IAM....208
10.2.2 Dependencies and Agencies....216
10.2.3 Workspace....253
10.3 Configuring Agency Authorization for ModelArts with One Click....253
10.4 Creating an IAM User and Granting ModelArts Permissions....260
10.5 Creating and Managing Workspaces....268
10.6 Configuration Practices in Typical Scenarios....273
10.6.1 Assigning Basic Permissions for Using ModelArts....273
10.6.1.1 Scenario....273
10.6.1.2 Step 1 Creating a User Group and Adding Users to the User Group....276
10.6.1.3 Step 2 Assigning Permissions for Using Cloud Services....276
10.6.1.4 Step 3 Configuring Agent-based ModelArts Access Authorization....278
10.6.1.5 Step 4 Verifying User Permissions....279
10.6.2 Assigning Basic Permissions for Using Development Environments to an IAM User....279
10.6.3 Assigning Basic Permissions for Using New-Version Inference to an IAM User....290
10.6.4 Separately Assigning Permissions to Administrators and Developers....292
10.6.5 Assigning the Required Permissions....296
10.6.6 Logging In to a Training Container Using Cloud Shell....298
10.6.7 Prohibiting a User from Using a Public Resource Pool....300
10.6.8 Authorizing ModelArts to Use SFS Turbo....302

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

10.6.9 Assigning SFS Turbo Folder-Level Access Permissions to an IAM User..... 305
10.7 FAQ..... 309
10.7.1 What Do I Do If a Message Indicating Insufficient Permissions Is Displayed When I Use ModelArts?..... 309
11 Best Practices of Security Configuration..... 313

1 ModelArts Best Practices

This document provides ModelArts samples concerning a variety of scenarios and AI engines to help you quickly understand the process and operations of using ModelArts for AI development.

NOTE

All practical case studies provided in this document are intended solely for scenarios such as function validation and prototype testing. Given the stringent requirements of production environments for high availability, security compliance, observability, and operations management, do not use the case studies in this document directly in production environments.

LLM Training and Inference

Sample	Scenario	Description
Deploying a Model from the vLLM-Ascend Open-Source Community on the ModelArts Inference Platform	Inference deployment	Describes how to deploy models obtained from the vLLM-Ascend open-source community on the ModelArts inference platform.

Video Generation Model Training and Inference

Sample	Scenario	Description
Inference Guide for Wan2.1, Wan2.2, HunyuanVideo, and CogVideo Series Models Adapted to NPU via ModelArts Lite Server (6.5.911)	Wan series model inference	Describes the inference process of Wan series models based on ModelArts Lite Server. The PyTorch framework and Ascend NPUs are used for inference.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

ModelArts Development Environment Cases

Table 1-1 ModelArts samples

Sample	Function	Scenario	Description
Migrating the Conda Environment on a Notebook Instance to an SFS Disk	Environment migration	Development environments	Describes how to migrate the Conda environment of a notebook instance to an SFS disk.

ModelArts Inference Deployment

Table 1-2 Inference deployment samples

Sample	Scenario	Description
Accessing a Real-Time Service Through a Dedicated API Gateway, WAF, VPC, and ELB	Inference deployment	Describes how to use Dedicated API Gateway (APIG), Web Application Firewall (WAF), Virtual Private Cloud (VPC), and Elastic Load Balance (ELB) to access ModelArts real-time services over the intranet.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

2 LLM Inference

2.1 Deploying a Model from the vLLM-Ascend Open-Source Community on the ModelArts Inference Platform

Based on the vLLM-Ascend inference framework and the real-time service deployment capabilities of the ModelArts inference platform, open-source models can be quickly deployed and used for inference in Ascend NPU scenarios. This is applicable to inference and deployment scenarios of open-source LLMs such as GLM, Qwen, and DeepSeek.

NOTE

This deployment guide uses models from the vLLM-Ascend open-source community. It is meant for function verification and prototype testing only. Given the stringent requirements of production environments for high availability, security compliance, observability, and operations management, do not use this solution directly in production environments. The model's capabilities and performance rely on the open-source community.

Use Case

When you need to deploy LLMs from open-source communities such as Hugging Face and ModelScope to the Ascend NPU environment of the ModelArts inference platform, you may face the following challenges: large model weights, adaptation of the inference framework to the Ascend CANN operator, configuration of tensor parallelism and data parallelism parameters for multi-node multi-PU deployment, and inconsistent service status caused by improper health check configuration. Based on the vLLM-Ascend inference framework and ModelArts real-time service deployment capabilities, you can obtain models, prepare images, configure scripts, and deploy services in one-stop mode. This greatly simplifies the deployment of open-source models in Ascend environments.

Typical scenarios include:

- Ascend NPU-based model inference: Open-source LLMs such as GLM, Qwen, and DeepSeek are deployed on Ascend devices to provide text generation and Q&A capabilities.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

- Multi-node multi-PU parallel inference: The tensor parallelism (TP) and data parallelism (DP) policies are used to split a large model for deployment on multiple NPUs, breaking the single-PU memory limit.
- Quantized model inference: Quantized models such as W8A8 are deployed to reduce memory usage and inference latency, while improving throughput.
- Service-oriented general-purpose model inference: Any open-source model supported by vLLM-Ascend is encapsulated into an OpenAI-compatible API for service systems to call.

Solution Process

To obtain a model from an open-source community and deploy it on the ModelArts inference platform, perform the following steps:

1. Prepare model weights: Download model weights from ModelScope or Hugging Face and upload them to an OBS bucket.
2. Prepare the deployment image: Download the vLLM-Ascend image from Quay.io and push it to an SWR image repository.
3. Prepare the startup script: Compile the startup script and health check script based on the model adaptation document and upload them to an OBS bucket.
4. Deploy the inference service: Configure a real-time service on ModelArts, mount the model weights and startup script, and start the inference service.
5. Verify the service: Call the inference service via APIs to verify service availability.

Constraints

- The model must be from the vLLM-Ascend official **model list**. Otherwise, you need to verify the model adaptation.
- During multi-node deployment, you need to ensure that the network communication is normal, as HCCL communication depends on the network between nodes.

Resource and Cost Planning

Table 2-1 lists the resources and cost planning used in this case.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Table 2-1 Resources and costs

Resource	Description	Quantity	Billing
ModelArts dedicated resource pools	NPUs are required for real-time service deployment. GLM-5.2-W8A8 dual-node deployment is used as an example. • **Resource Pool Type**: Select **Physical**. • **Flavor Type**: Select **NPU**. • **Flavor**: Select **8 * ascend-snt9b (A2)**. • **Instances**: Set it to **2**.	1	Pay-per-use billing. The fee depends on the NPU specifications and usage duration.
Object Storage Service (OBS)	Stores model weights, startup scripts, and health check scripts. Set **Storage Class** to **Standard**.	1	Billed by storage capacity and number of requests.
Software Repository for Container (SWR)	Hosts vLLM-Ascend inference images.	1	Billed by image storage capacity and traffic.
Elastic Cloud Server (ECS)	Loads images and pushes them to SWR. • During image creation, if interaction with websites such as GitHub and Hugging Face is involved, you are advised to purchase ECSs at CN-Hong Kong to avoid image creation failures due to network proxy problems. • Set **CPU Architecture** to **Kunpeng** and **Image** to **EulerOS**, and bind an EIP. • There must be no less than 50 GB available space.	1	Billed by node specifications and usage duration.

Preparations

Before the deployment, ensure that you have:

- Enabled ModelArts, OBS, and SWR.
- Created a ModelArts dedicated resource pool with Snt9b NPU resources.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

- Created an OBS bucket and an SWR organization, which are in the same region as ModelArts.
- Created an ECS. For details, see **Purchasing and Using a Linux ECS**.

Step 1: Download the Model Weight and Upload It to OBS

1. Access the vLLM-Ascend model adaptation description list, confirm that the target model is supported, and check the model weight requirements.

   GLM-5.2-W8A8 adaptation document: **https://docs.vllm.ai/projects/ascend/en/latest/tutorials/models/GLM5.2.html**.

2. Download the model weight from ModelScope or Hugging Face.

   Download the GLM-5.2-W8A8 model weight from ModelScope: **https://www.modelscope.cn/models/Eco-Tech/GLM-5.2-w8a8/files**

   CAUTION

   Ensure that the downloaded model weight file is complete and contains the weight file (**.safetensors**), configuration file (**config.json**), and **tokenizer**.

3. Upload the downloaded model weight to the **models** directory in the OBS bucket.

   OBS path planning for the model weight file:

   ```
   ├── <OBS-bucket>
   │   └── models
   │       └── <model-name>              # Directory where the model weight is located, which is mounted to the container during deployment.
   ```

   NOTE

   It takes a long time to upload large model weight files to OBS. Models with tens of billions of parameters are about 20 GB, and models with hundreds of billions of parameters are about 200 GB. The upload time depends on the network bandwidth.

Step 2: Download the Inference Image and Push It to SWR

Download the vLLM-Ascend image from the official image repository and push it to SWR through an ECS node.

1. Choose the inference image required by the target model based on the vLLM-Ascend model adaptation description.

   Take GLM-5.2 as an example, the inference image name is **quay.io/ascend/vllm-ascend:glm5.2** for the A2 series.

2. Remotely log in to the ECS and check whether Docker has been installed.

   ```
   # Check whether Docker is installed.
   docker -v
   ```

   If Docker is not installed, run the following command:

   ```
   yum install -y docker-engine.aarch64 docker-engine-selinux.noarch docker-runc.aarch64
   ```

3. Use Docker to download the image on the ECS.

   ```
   docker pull --platform linux/arm64 <inference-image-name>
   ```

   The inference image name must be that obtained in **1**.

   Example:

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
docker pull --platform linux/arm64 quay.io/ascend/vllm-ascend:glm5.2
```

CAUTION

If an Arm-powered image is used by the Ascend device, you must specify **--platform linux/arm64** when downloading the image. Otherwise, the inference fails due to image architecture mismatch.

If the download is successful, the download status is displayed:

```
Status: Downloaded newer image for quay.io/ascend/vllm-ascend:glm5.2
```

Troubleshooting

- If the **--platform** parameter is unavailable when you run the **pull** command, rectify the fault by referring to **FAQ: What Can I Do If "The --platform Parameter Is Not Supported" Is Displayed When I Run the pull Command?**.
- If ECS cannot connect to Quay.io when you run the **pull** command, rectify the fault by referring to **FAQ: What Can I Do If the pull Respond Times Out Because the ECS Network Cannot Access the Image Repository?**.

4. Log in to SWR, choose **My Images**, and click **Upload Through Client** in the upper right corner. In the displayed dialog box, click **Generate Login Command** to obtain the Docker login command.

   Login command example:

   ```
   docker login -u <region>@<account-AK> -p <temporary-token> <image-address>
   ```

   For details, see **Pushing an Image**.

5. Enter the obtained login command on the ECS to log in to Docker.

   If **Login Succeeded** is returned, the login is successful.

6. Tag the inference image.

   ```
   docker tag <inferenc-image-name> <image-address>/<organization-name>/<image-name>:<version>
   ```

Table 2-2 Tag parameters

Parameter	Description
Inference Image Name	Actual image version to be pushed to SWR, for example, **quay.io/ascend/vllm-ascend:glm5.2**.
Image Address	SWR image address, that is, the domain name at the end of the login command.
Organization Name	SWR organization name created in preparations, for example, **modelarts**.
Image Name	Custom image name, which can be the same as the actual image name, for example, **vllm-ascend**.
Version	Custom version, which can be the same as the actual image version, for example, **glm5.2**.

Example:

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
docker tag quay.io/ascend/vllm-ascend:glm5.2 swr.xxx.example.com/modelarts/vllm-ascend:glm5.2
```

7. Push the tagged inference image to SWR.

```
docker push <image-address>/<organization-name>/<image-name>:<version>
```

Example:

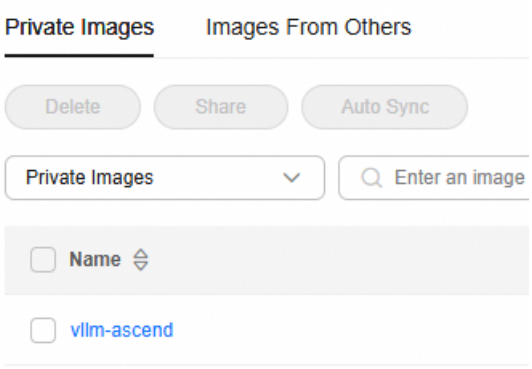
```
docker push swr.xxx.example.com/modelarts/vllm-ascend:glm5.2
```

If the push is successful, the push progress of each layer and the final digest are displayed.

```
glm5.2: digest: xxx:xxxxx size: xxxx
```

8. Check whether the image is uploaded on the SWR console. To do so, choose **My Images** > **Private Images**, and check whether the image has been pushed.

Figure 2-1 Checking the image



Step 3: Compile a Script and Upload It to OBS

1. Confirm the startup command requirements of the target model based on the vLLM-Ascend model adaptation description.

   GLM-5.2-W8A8 adaptation document: **https://docs.vllm.ai/projects/ascend/en/latest/tutorials/models/GLM5.2.html**

2. Compile the **get_master_ip.py** tool for obtaining the master IP address from the global rank table. This tool is used to automatically detect the master/slave role during multi-node PD deployment. During single-node deployment, that is, there is only one inference unit (DP = 1), there is no need to distinguish the master and slave roles. Therefore, this script is not required.

```
import json
import os
import time
import sys

GLOBAL_RANK_TABLE_ENV = 'GLOBAL_RANK_TABLE_FILE_PATH'

def wait_and_get_completed_global_rank_table():
    global_rank_table_path = os.getenv(GLOBAL_RANK_TABLE_ENV)
    if not global_rank_table_path:
        print('read env "{}" failed'.format(GLOBAL_RANK_TABLE_ENV))
        return
    while True:
        try:
            rank_table = get_rank_table(global_rank_table_path)
            if rank_table is not None:
                return rank_table
        except Exception as e:
            print(e)
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
        time.sleep(1)

def get_rank_table(global_rank_table_path=None):
    if global_rank_table_path is None:
        global_rank_table_path = os.getenv(GLOBAL_RANK_TABLE_ENV)
    with open(global_rank_table_path, 'r') as file:
        buf = file.read()
    rank_table = json.loads(buf)
    if rank_table["status"] == "completed":
        return rank_table
    else:
        return None

def get_master_ip():
    try:
        wait_and_get_completed_global_rank_table()
        global_rank_table_path = os.getenv(GLOBAL_RANK_TABLE_ENV)
        with open(global_rank_table_path, 'r') as file:
            buf = file.read()
        rank_table = json.loads(buf)
        master = rank_table['server_group_list'][0]['server_list'][0]
        return master['server_ip']
    except Exception as e:
        print(e)

def get_dp_start_rank():
    try:
        wait_and_get_completed_global_rank_table()
        global_rank_table_path = os.getenv(GLOBAL_RANK_TABLE_ENV)
        pod_ip = os.getenv("POD_IP")
        if not pod_ip:
            print("read env POD_IP failed")
            return 0
        with open(global_rank_table_path, 'r') as file:
            buf = file.read()
        rank_table = json.loads(buf)
        rank = 0
        for group in rank_table.get('server_group_list', []):
            for server in group.get('server_list', []):
                if server.get('server_ip') == pod_ip or server.get('server_id') == pod_ip or
server.get('container_ip') == pod_ip:
                    return rank
                rank += 1
        print(f"Warning: POD_IP {pod_ip} not found in rank table, fallback to rank 0")
        return 0
    except Exception as e:
        print(f"get_dp_start_rank error: {e}")
        return 0

if __name__ == "__main__":
    if len(sys.argv) > 1 and sys.argv[1] == "--dp-rank":
        print(get_dp_start_rank())
    else:
        print(get_master_ip())
```

3. Compile the model startup script. For details about the script content, see the deployment part in the vLLM-Ascend model adaptation description. The following provides script examples in different deployment modes.
   - Multi-node PD deployment: The master/slave role needs to be detected. The following uses **glm_5_2.sh** (the model startup script used in this case) as an example.

     ```
     #!/bin/sh
     CUR_DIR=$(cd "$(dirname "$0")" && pwd)

     # ============ Model-specific parameters ============
     MODEL_PATH="/weight/"
     SERVED_MODEL_NAME="glm-52"
     PORT="7000"
     ```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
TENSOR_PARALLEL_SIZE=8
DATA_PARALLEL_SIZE=2
DATA_PARALLEL_SIZE_LOCAL=1
MAX_MODEL_LEN=40000
MAX_NUM_BATCHED_TOKENS=4096
GPU_MEMORY_UTILIZATION=0.95
MAX_NUM_SEQS=16
QUANTIZATION="ascend"
EXTRA_ARGS="--enable-expert-parallel \
  --safetensors-load-strategy 'prefetch' \
  --block-size 128 \
  --async-scheduling \
  --additional-config '{\"fuse_muls_add\": true, \"multistream_overlap_shared_expert\": true,
\"ascend_compilation_config\": {\"enable_npugraph_ex\": true}}' \
  --compilation-config '{\"cudagraph_mode\": \"FULL_DECODE_ONLY\"}' \
  --speculative-config '{\"num_speculative_tokens\": 5, \"method\": \"deepseek_mtp\"}'"

# Network settings
nic_name=$(ifconfig | grep -B1 "$POD_IP" | grep -o "^\w*")
master_ip=$(PYTHONUNBUFFERED=1 python3 "$CUR_DIR"/get_master_ip.py)
dp_start_rank=$(PYTHONUNBUFFERED=1 python3 "$CUR_DIR"/get_master_ip.py --dp-rank)
echo "master ip:${master_ip}, local ip: ${POD_IP}, nic:${nic_name}, dp_rank:${dp_start_rank}"

# Determine the node role.
if [ "$POD_IP" = "$master_ip" ]; then
    echo "Current node is MASTER"
    slave_opts="--api-server-count 1"
else
    echo "Current node is SLAVE"
    nohup python -m http.server 8080 > http_server.log 2>&1 &
    slave_opts="--headless"
fi

# Environment variable configuration
export HCCL_OP_EXPANSION_MODE="AIV"
export HCCL_IF_IP=$POD_IP
export GLOO_SOCKET_IFNAME=$nic_name
export TP_SOCKET_IFNAME=$nic_name
export HCCL_SOCKET_IFNAME=$nic_name
export VLLM_RPC_TIMEOUT=360000
export VLLM_EXECUTE_MODEL_TIMEOUT_SECONDS=3000
export HCCL_EXEC_TIMEOUT=200
export HCCL_CONNECT_TIMEOUT=120
export OMP_PROC_BIND=false
export OMP_NUM_THREADS=10
export PYTORCH_NPU_ALLOC_CONF=expandable_segments:True
export ACL_OP_INIT_MODE=1
export TASK_QUEUE_ENABLE=1
export CPU_AFFINITY_CONF=1
export VLLM_ENGINE_READY_TIMEOUT_S=1200

# Build quantization parameters (optional, only for quantized models).
quant_opts=""
if [ -n "$QUANTIZATION" ]; then
    quant_opts="--quantization ${QUANTIZATION}"
fi

# Start the service.
vllm serve ${MODEL_PATH} \
  --host 0.0.0.0 \
  --port ${PORT} \
  --max_model_len ${MAX_MODEL_LEN} \
  --max-num-batched-tokens ${MAX_NUM_BATCHED_TOKENS} \
  --served-model-name ${SERVED_MODEL_NAME} \
  --seed 1024 \
  --gpu-memory-utilization ${GPU_MEMORY_UTILIZATION} \
  --max-num-seqs ${MAX_NUM_SEQS} \
  --data-parallel-size ${DATA_PARALLEL_SIZE} \
  --data-parallel-size-local ${DATA_PARALLEL_SIZE_LOCAL} \
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
  --data-parallel-start-rank ${dp_start_rank} \
  --data-parallel-address ${master_ip} \
  --data-parallel-rpc-port 13389 \
  --tensor-parallel-size ${TENSOR_PARALLEL_SIZE} \
  ${quant_opts} ${EXTRA_ARGS} \
  $slave_opts
```

– Single-node deployment: The master/slave role does not need to be distinguished. The model startup script is simpler. The following uses **minitron-8b-base.sh** as an example.

```
#!/bin/sh

# ============ Model-specific parameters ============
MODEL_PATH="/weight/"
SERVED_MODEL_NAME="minitron-8b-base"
PORT="8000"
TENSOR_PARALLEL_SIZE=1
MAX_MODEL_LEN=4096
GPU_MEMORY_UTILIZATION=0.9
EXTRA_ARGS="--enforce-eager"

# Start the service.
vllm serve ${MODEL_PATH} \
  --host 0.0.0.0 \
  --port ${PORT} \
  --max_model_len ${MAX_MODEL_LEN} \
  --served-model-name ${SERVED_MODEL_NAME} \
  --gpu-memory-utilization ${GPU_MEMORY_UTILIZATION} \
  --tensor-parallel-size ${TENSOR_PARALLEL_SIZE} \
  ${EXTRA_ARGS}
```

Table 2-3 Startup script parameters

Variable	Description	Example Value
MODEL_PATH	Model weight mounting path, which must be the same as that on ModelArts.	/weight/
SERVED_MODEL_NAME	Inference service name, which is used during API calling. It is recommended that the value be the same as the model name.	glm-52
TENSOR_PARALLEL_SIZE	Number of TPs, which must be the same as the number of NPUs used by each inference unit during deployment.	8
DATA_PARALLEL_SIZE	Number of DPs, which must be the same as the total number of instances used during deployment (Total number of instances = Number of inference units x Number of instances per inference unit).	2
DATA_PARALLEL_SIZE_LOCAL	Number of DPs on a single node, which must be the same as the number of instances per inference unit during deployment.	1

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Variable	Description	Example Value
MAX_MODEL_LEN	Maximum context length, which is configured based on the vLLM-Ascend model adaptation description.	40,000
MAX_NUM_BATCHED_TOKENS	Maximum number of tokens per batch, which can be configured as required.	4,096
GPU_MEMORY_UTILIZATION	Memory usage, which is configured based on the vLLM-Ascend model adaptation description.	0.95
MAX_NUM_SEQS	Maximum number of concurrent sequences, which can be adjusted based on service concurrency requirements.	16
QUANTIZATION	Quantization mode. For W8A8 models, **ascend** is used. For non-quantized models, this variable can be deleted.	ascend
EXTRA_ARGS	Model-specific optimization parameters, including expert parallelism, speculative decoding, and operator fusion. Replace or delete the parameters based on the model adaptation document.	--enable-expert-parallel ...
PORT	Listening port of the vLLM inference service. The value must be the same as the container port during deployment. Ensure that there is no port conflict.	7000

4. Compile the **check.sh** health check script. The following provides script examples for different deployment modes. The value of **PORT** must be the same as that in the model startup script.
   - Multi-node PD deployment: You need to call **get_master_ip.py** (health check script used in this case) to obtain the master IP address.

```
#!/bin/bash
PROBE_TYPE=$1

# The port number must be the same as that in the model startup script.
PORT="7000"

if [[ $PROBE_TYPE == "startup" ]]; then
    if pgrep -f "vllm serve" > /dev/null; then
    exit 0
    fi
    exit 1
fi
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
master_ip=$(PYTHONUNBUFFERED=1 python3 /code/get_master_ip.py)

if [[ $PROBE_TYPE == "readiness" ]]; then
    HEALTH_URL="http://${master_ip}:{PORT}/health"
    status_code=$(curl -s -o /dev/null -w "%{http_code}" $HEALTH_URL)
    if [ "$status_code" -eq 200 ]; then
      exit 0
    else
      echo "readiness check failed" >&2
      exit 1
    fi
fi

if [[ $PROBE_TYPE == "liveness" ]]; then
    if pgrep -f "vllm serve" > /dev/null; then
    exit 0
    fi
    exit 1
fi
```

– Single-node deployment: The master/slave role does not need to be distinguished. The health check script is simpler.

```
#!/bin/bash
PROBE_TYPE=$1

# The port number must be the same as that in the model startup script.
PORT="8000"

if [[ $PROBE_TYPE == "startup" ]]; then
    if pgrep -f "vllm serve" > /dev/null; then
    exit 0
    fi
    exit 1
fi

if [[ $PROBE_TYPE == "readiness" ]]; then
    HEALTH_URL="http://${POD_IP}:${PORT}/health"
    status_code=$(curl -s -o /dev/null -w "%{http_code}" $HEALTH_URL)
    if [ "$status_code" -eq 200 ]; then
      exit 0
    else
      echo "readiness check failed" >&2
      exit 1
    fi
fi

if [[ $PROBE_TYPE == "liveness" ]]; then
    if pgrep -f "vllm serve" > /dev/null; then
    exit 0
    fi
    exit 1
fi
```

5. Upload the startup script and health check script to the **code** directory of the OBS bucket.

   OBS path planning for the script file:

```
├── <OBS-bucket>
│   └── models
│       └── <model-name>                # Directory where the model weight is located, which is mounted to the container during deployment.
│   └── code
│       ├── get_master_ip.py            # Obtain the master IP tool from the global rank table (only for multi-node PD deployment).
│       ├── glm_5_2.sh                  # Model startup script
│       └── check.sh                    # Health check script
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Step 4: Deploy the Inference Service

Configure and start a real-time service on ModelArts.

1. Log in to the **ModelArts console**.
2. (Recommended) Warm up the model weight in the dedicated resource pool in advance for faster deployment.

   NOTE

   For first-time deployment of the inference service, you need to download the model weight from OBS to the node. You can load the model in advance using resource pool warmup as it takes a long time to load large models.

   a. In the navigation pane on the left, choose **Resource Management** > **Dedicated Compute Resources** > **Resource Pools**.
   b. Click the target dedicated resource pool to access its details page.
   c. In the **Warmup** tab, click **Add Warmup Task**. In the displayed dialog box, set the parameters. The table below lists only key parameters. For details, see **Adding a Model Warmup Task to a Dedicated Resource Pool**.

   Table 2-4 Key parameters for adding a model warmup task

Parameter	Description	Example Value
Weight Path	Select the OBS path of the model weight file to be warmed up, that is, the path in **Step 1: Download the Model Weight and Upload It to OBS**.	obs://xxx/models/
File Space (GB)	Enter the size of the model weight file to be warmed up. After **Weight Path** is configured, click **Obtain weight file size** to obtain the recommended value. Generally, the predicted value is slightly greater than the actual size of the model directory.	780

   d. After the configuration is complete, click **OK** to submit the model warmup task.

      When **Status** changes to **Ready**, the model warmup is complete.
3. Return to the ModelArts main menu and choose **Model Inference** > **Real-Time Inference** from the navigation pane. The real-time service management page is displayed.
4. Click **Deploy** in the upper right corner. The **Deploy Real-Time Service** page is displayed.
5. On the displayed page, configure service information. The table below only lists key parameters. For details, see **Configuring Service Information**.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Table 2-5 Key parameters of service information

Category	Parameter	Description	Example Value
Basic Information	Service Name	Custom service name. You are advised to add the model name to this value.	glm_52_w8a8
Network Settings	Service Access Method	Select **Default** to use ModelArts capabilities.	Default
	Service Protocol	Select **HTTP** and set the request timeout interval as required.	HTTP
	Authentication Mode	Select an authentication mode as required. In the test environment, you can select **None** to directly call APIs.	None

6. Click **Next** to configure service deployment. The table below only lists key parameters. For details, see **Deploying a Real-Time Inference Service Using Multi-Node PD Co-location**.

Table 2-6 Key parameters for deployment configuration

Category	Parameter	Description	Example Value
Resource Settings	Resource Pool Type	Select **Dedicated Resource Pool** to obtain exclusive resources.	Dedicated Resource Pool
Model Settings	Model Source	Select **Custom Model**.	Custom Model
	Storage Type	Select **Pre-warmed model**.	Pre-warmed model
	Pre-warmed Model	Select a pre-warmed model.	glm52w8a8
	Mount Path	Mount path of the model weight in the container, which must be the same as that in the startup script **glm_5_2.sh**.	/weight/

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Category	Parameter	Description	Example Value
Unit Settings	Deployment Mode	Select a model deployment mode. • **Basic mode**: Single-node deployment. An inference unit is configured. Role detection and PD communication are not required. • **Multi-role**: Multi-node PD deployment. Multiple inference units are configured. The **get_master_ip.py** script is required to check the master and slave roles.	Multi-role
	Unit Name	Customized inference unit name.	role-0
	Specification Type	Select **Custom**.	Custom
	NPU	Number of NPUs used by each inference unit, which must be the same as the value of **TENSOR_PARALLEL_SIZE** in the model startup script, for example, **glm_5_2.sh**.	8
	CPU	Set this parameter based on the resource pool specifications.	144
	Memory	Set this parameter based on the resource pool specifications.	512,000 MB
	Single-Replica Instances	Number of instances on each inference unit, which must be the same as the value of **DATA_PARALLEL_SIZE_LOCAL** in the model startup script, for example, **glm_5_2.sh**.	1
	Image Type	Select **Custom Images**.	Custom Images
	Select Image	Select a private image from SWR, that is, the image in **Step 2: Download the Inference Image and Push It to SWR**.	vllm-ascend:glm52
	Mount File Storage	Mount the **code** folder in OBS to the container. Under **File Storage**, set **Storage Type** to **Object Storage Service — Bucket**, **Storage Location** to **code**, and **Mount Path** to **/code/**.	See the parameter description.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Category	Parameter	Description	Example Value
	Health Check	Configure a health check. • **Startup Probe**: Check whether the vLLM process is started. Set **Check Method** to **Command** and enter **bash /code/check.sh startup**. • **Readiness Probe**: Check whether the inference service can receive requests (only the /health API on the master node is checked). Set **Check Method** to **Command** and enter **bash /code/check.sh readiness**. • **Liveliness Probe**: Check whether the vLLM process is running. Set **Check Method** to **Command** and enter **bash /code/check.sh liveness**.	See the parameter description.
	Boot Command	Enter the boot command and mount the **/code/** directory. After the container is started, the code path in OBS is mounted to this path. In the boot command, use the boot script in this path to start the vLLM service.	bash /code/glm_5_2.sh
	Add Inference Unit	During multi-node PD deployment, add inference units as required and configure the parameters by referring to the first inference unit. Customize the unit name, for example, **role-1**. Set same values for other parameters.	-
Deployment Management Settings	Container Protocol	Network transmission protocol of the container.	HTTP
	Container Port	Listening port of the image, which must be the same as the value of **port** in the model startup script, for example, **glm_5_2.sh**.	7000

7. Click **Next** and confirm the configuration details.
8. Click **Deploy** to start the real-time service.
9. When the inference service is in the **Running** state, click the service name to access its details page, and click the **Logs** tab.
10. In the displayed tab, check the deployment log. If the information below is displayed, the inference service is deployed.
    ```
    INFO: Application startup complete.
    ```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Step 5: Verify Service Availability via APIs

Verify the availability of the inference service via APIs.

1. Click the deployed inference service in the list to access its details page.
2. Click the **Prediction** tab.
3. Set the request parameters.

   Example:

   - Request method: POST
   - Request URL: /v1/completions
   - Data format: stream
   - Request body:

     ```
     {
        "model": "glm-52",
        "prompt": "The future of AI is",
        "max_completion_tokens": 50,
        "temperature": 0
     }
     ```

4. Click **Predict** and view the returned result. If the service is normal, the model name, model information, and answer are returned.

   For the first request, the model needs to be compiled and the cache needs to be warmed up. It is normal that the response is slow. Subsequent requests are responded to normally.

FAQ: What Can I Do If "The --platform Parameter Is Not Supported" Is Displayed When I Run the pull Command?

If the error below is reported when you run the **pull** command, you need to enable the Docker experiment feature before using **--platform**.

```
"--platform" is only supported on a Docker daemon with experimental features enabled
```

Enable the experiment mode:

1. Edit **/etc/docker/daemon.json**:

   ```
   {
     "experimental": true
   }
   ```

2. Restart Docker:

   ```
   systemctl restart docker
   ```

3. Run the **pull** command again.

FAQ: What Can I Do If the pull Respond Times Out Because the ECS Network Cannot Access the Image Repository?

If the error below is reported when you run the **pull** command, ECS cannot connect to Quay.io. In this case, download the image from a server that can connect to Quay.io, export the image, and upload it to the current ECS.

```
Error response from daemon: Get "https://quay.io/v2/": net/http: request canceled while waiting for connection (Client.Timeout exceeded while awaiting headers)
```

Solution:

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

1. Check the network between the ECS and Quay.io. If a timeout error is reported, the ECS is isolated from the network and cannot directly access the image repository.
   ```
   curl -I https://quay.io
   ```
2. Check the model deployment environment.
   - If the model is to be deployed in a LAN, go to the next step. Use another ECS that supports public network access to download the image, export the image, and upload it to the ECS in the LAN.
   - If the model is to be deployed on a public network, bind an EIP to the current ECS and perform **Step 2: Download the Inference Image and Push It to SWR** again to continue the deployment.
3. Use another ECS that supports public network access to download the image.
   a. Remotely log in to the ECS and check whether Docker has been installed.
      ```
      # Check whether Docker is installed.
      docker -v
      ```
      If Docker is not installed, run the following command:
      ```
      yum install -y docker-engine.aarch64 docker-engine-selinux.noarch docker-runc.aarch64
      ```
   b. Use Docker to download the image on the ECS.
      ```
      docker pull --platform linux/arm64 <inference-image-name>
      ```
      The inference image name must be that obtained in **1**.

      Example:
      ```
      docker pull --platform linux/arm64 quay.io/ascend/vllm-ascend:glm5.2
      ```
4. After the image is downloaded, export it as a **vllm-ascend.tar** file.
   ```
   docker save <inference-image-name> > vllm-ascend.tar
   ```
   Example:
   ```
   docker save quay.io/ascend/vllm-ascend:v0.22.1rc1 > vllm-ascend.tar
   ```
5. Upload **vllm-ascend.tar** to the ECS in the LAN. You can use physical transmission methods such as SCP, FTP, portable hard disk, or USB flash drive, or use OBS obsutil to upload and download data.
6. On the ECS in the LAN, load the image to the local Docker.
   ```
   docker load < vllm-ascend.tar
   ```
   If the operation is successful, the name of the loaded image is displayed.
   ```
   Loaded image: quay.io/ascend/vllm-ascend:v0.22.1rc1
   ```
7. Push the loaded image to SWR on the intranet. For details, see **4** to **8**.
8. Continue with the model deployment.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

3 LLM Training

3.1 Training the Qwen3 Series Models Adapted to the MindSpeed-LLM Framework on ModelArts

3.1.1 Overview

Description

ModelArts allows you to fine-tune popular open-source models with the MindSpeed-LLM framework.

MindSpeed-LLM

MindSpeed-LLM is a distributed training suite for LLMs based on the Ascend ecosystem. It aims to provide an E2E LLM training solution for Huawei **Ascend chip** ecosystem partners, including distributed pre-training, distributed instruction fine-tuning, and the corresponding development toolchain, such as data preprocessing, weight transformation, online inference, and baseline evaluation.

Supported LLMs

Table 3-1 Supported LLMs

Model	Training Scenario	Megatron	Training Framework	Minimum PUs
Qwen3-8B	SFT	√	MindSpeed-LLM	8x Snt9b2
Qwen3-30B-A3B	SFT	√	MindSpeed-LLM	16 x Snt9b2
Qwen3-32B	SFT	√	MindSpeed-LLM	16 x Snt9b2

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Constraints

- Snt9b2 is used for training.

Resources and Files Required

Table 3-2 Resources and data files required for training

No.	Item	Mandatory	Description
1	ModelArts compute resources	Yes	You are advised to use Snt9b resources in a ModelArts dedicated resource pool. The pool must be created in advance.
2	OBS bucket	Yes	ModelArts requires you to create an OBS bucket for storing the model weight files, training data, training scripts, and training artifacts.
3	Model weight files	Yes	Model files for training. You must prepare them in advance. This document provides the download addresses.
4	Training datasets	Yes	Datasets for training. You must prepare them in advance. This document provides the download addresses.
5	Training scripts	Yes	Dataset preprocessing scripts and training job startup scripts. You must prepare them in advance. This document provides script examples, which can be directly used.
6	Training environment image	No	The training images are already set up on the platform. You do not need to prepare them. Just select an image when you create a training job. The image contains the software required for training, such as the CANN package, driver, PyTorch, and training framework code package.

Procedure

Table 3-3 Procedure description

Phase	Step	Description
Preparations	Configuring Cloud Resources	Prepare the dedicated resource pool and OBS bucket required for training.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Phase	Step	Description
	Preparing Model Weights and Training Data	Prepare the model weight files, training datasets, and data processing script files, and upload them to the OBS bucket.
	Preparing the Training Scripts	Prepare the script files required for training in advance and upload them to the OBS bucket.
Training	Executing a Training Job	Create a training job. During the creation, select a preset training environment image, select the resource specifications required for training, and set input and output parameters.
Viewing the training results	Viewing the Training Output	View the training logs and performance results.
	Viewing the Training Output	Obtain the trained model from the output path set in the training job.

3.1.2 Configuring Cloud Resources

The resources required for this guide are listed below. Prepare them in advance.

Resource Requirements

Table 3-4 Resources to be prepared

Item	Mandatory	Description	Reference
ModelArts compute resources	Yes	You are advised to use Snt9b resources in ModelArts dedicated resource pools. The dedicated resource pool must be created in advance. In addition, the dedicated resource pool must be able to access the external network through the VPC.	For details, see Creating a Dedicated Resource Pool.
OBS bucket	Yes	ModelArts requires you to create an OBS bucket for storing the model weight files, training data, training scripts, and training artifacts.	Creating an OBS Bucket

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Creating an OBS Bucket

ModelArts requires you to create an OBS bucket for storing the model weight files, training data, training scripts, and training artifacts. For details, see **How Do I Create an OBS Bucket for ModelArts to Store Data?**

The following shows the folders in the OBS bucket. The planned directories must be the same as the file paths involved in the training script. After a training job is created and the OBS bucket is added to the code directory, all files in the folder will be copied to the training container. If the file size is too large, the copy speed will decrease. You are advised to only retain related files in the directory to avoid storing irrelevant content. For example, for each training job, the *{models}* folder only stores the models used this time.

```
|——{OBS bucket}                          # Custom bucket name, for example, mindspeed-llm
  |——{mindspeed-a2}                      # Custom OBS folder name, for example, mindspeed-llm-a2
    |——{models}                          # Custom OBS folder for storing various models, for example, models
      |——{Qwen3-8b}                      # Custom OBS folder for storing models, for example, Qwen3-8b
      |——{Qwen3-30b-a3b}                 # Custom OBS folder for storing models, for example, Qwen3-30b-a3b
      |——{Qwen3-32b}                     # Custom OBS folder for storing models, for example, Qwen3-32b
    |——ckpt_convert_qwen3_hf2mcore.sh    # Qwen3-32b weight conversion script (hf2mg)
    |——data_convert_qwen3_instruction.sh # Qwen3-32b data processing script
    |——tune_qwen3_32b_4K_full_ptd.sh     # Script for executing the Qwen3-32b fine-tuning task
    |——ckpt_convert_qwen3_mcore2hf.sh    # Qwen3-32b weight conversion script (mg2hf)
    |——run_distributed_task.sh           # Script for starting the complete Qwen3-32b fine-tuning task
    |——{dataset}/                        # Custom OBS folder for storing various datasets, for example, dataset.
      |——{alpaca}/                       # OBS folder for storing the training dataset alpaca
      |——{alpaca_gpt4_data}/             # OBS folder for storing the training dataset alpaca_gpt4_data
    |——{output}                          # Custom OBS folder for storing the final training output
```

You can use tools such as **OBS Browser+** and **obsutil** to access and manage OBS buckets and upload and download files and data.

3.1.3 Preparing Model Weights and Training Data

Obtaining the Model Weights and Code

This section uses Qwen models as an example to describe the training process. **Table 3-5** lists the supported LLMs and the addresses for obtaining their weights.

Table 3-5 Supported LLMs and model weight sources

Model	Training Scenario	Training Framework	Open-Source Weight File Link
Qwen3-8B	SFT	MindSpeed-LLM	**https://huggingface.co/Qwen/Qwen3-8B**
Qwen3-30B-A3B	SFT	MindSpeed-LLM	**https://huggingface.co/Qwen/Qwen3-30B-A3B**
Qwen3-32B	SFT	MindSpeed-LLM	**https://huggingface.co/Qwen/Qwen3-32B**

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

> **NOTE**
>
> To download weight files from the Hugging Face website, set up a proxy. For instructions, search online.

Upload the downloaded model weight files to your OBS bucket. The following is an example of the directory for storing files in your OBS bucket based on the OBS plan:

```
obs://mindspeed-llm/mindspeed-llm-a2/Qwen3-8B
obs://mindspeed-llm/mindspeed-llm-a2/Qwen3-32B
obs://mindspeed-llm/mindspeed-llm-a2/Qwen3-30B-A3B
```

Obtaining Training Data

The Qwen3 series models are used as an example. The following dataset can be used:

- Fine-tuning dataset

  alpaca_gpt4_data is used as an example.

  ```
  wget https://huggingface.co/datasets/QingyiSi/Alpaca-CoT/blob/main/alpacaGPT4/
  alpaca_gpt4_data.json
  ```

3.1.4 Preparing the Training Scripts

In MindSpeed-LLM, weight conversion, data processing, and fine-tuning task scripts vary across different model types. For specific script examples tailored to each model, see **MindSpeed-LLM/docs/quick_start.md · Ascend/MindSpeed-LLM - AtomGit | GitCode**.

For a Qwen3 training job, prepare the training script for the model and upload it to your OBS bucket. The scripts are as follows:

- **ckpt_convert_qwen3_hf2mcore.sh**: weight conversion script, which converts the weights from the hf format to the mg format. For details about the script example, see **Weight Conversion Script (hf2mg)**.
- **data_convert_qwen3_instruction.sh**: data processing script. For details about the script example, see **Data Processing Script**.
- **tune_qwen3_32b_4K_full_ptd.sh**: fine-tuning task execution script. For details about the script example, see **Fine-Tuning Task Execution Script**.
- **ckpt_convert_qwen3_mcore2hf.sh**: weight conversion script, which converts weights from the mg format to the hf format. For details about the script example, see **Weight Conversion Script (mg2hf)**.
- **run_distributed_task.sh**: training job startup script, which calls all the preceding scripts during execution. In the multi-node training scenario, the **run_distributed_task.sh** script is used to enable each node to obtain the IP address of each other and set **MASTER_PORT**. For details about the script example, see **Training Job Startup Script**.

Common Parameters in the Script

The parameters in the sh script are as follows:

- *${output_dir}*: output path of training or format conversion. Configure the OBS path for the **output_dir** environment variable. For example, **/home/ma-user/modelarts/outputs/output_dir_0**.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

- *${model_path}*: path of the training model. Set **TOKENIZER_PATH** based on this parameter.
- *${dataset_path}*: path of the dataset to be processed.

If you directly copy this script, pay attention to the differences between newline characters in different OSs. You can run the **dos2unix xxx.sh** command to convert the newline characters in the script to the newline characters in the Linux system.

Take **Qwen3-32B** as an example. Prepare the following script. The comments provided are for parameter explanation only; **remove** them before actual use. **Ensure there are no spaces after the backslash (\) at the end of each line**.

Weight Conversion Script (hf2mg)

The content of the **ckpt_convert_qwen3_hf2mcore.sh** script is as follows. For details about the parameters to be modified in the script, see the comments in the script.

The environment variables are set in the **run_distributed_task.sh** script for starting the training job.

```
# Change the ascend-toolkit path.
export CUDA_DEVICE_MAX_CONNECTIONS=1
source /usr/local/Ascend/ascend-toolkit/set_env.sh
python /home/ma-user/MA_Turbo/src/open_source/MindSpeed-LLM/convert_ckpt.py \ # Location of the MindSpeed-LLM convert code pre-installed in the image. No modification is required.
    --use-mcore-models \
    --model-type GPT \
    --load-model-type hf \
    --save-model-type mg \
    --target-tensor-parallel-size 8 \
    --target-pipeline-parallel-size 2 \
    --spec mindspeed_llm.tasks.models.spec.qwen3_spec layer_spec \
    --load-dir "$TOKENIZER_PATH" \                                  # Model location, which is set in run_distributed_task.sh.
    --save-dir "$CKPT_LOAD_DIR" \                  # Location for saving the converted weights, which is set in run_distributed_task.sh.
    --tokenizer-model "$TOKENIZER_PATH/tokenizer.json"  \             # Location of tokenizer.json. The variable part is set in run_distributed_task.sh.
    --params-dtype bf16 \
    --model-type-hf qwen3
```

Data Processing Script

The content of the **data_convert_qwen3_instruction.sh** script is as follows. For details about the parameters to be modified in the script, see the comments in the script.

The environment variables are set in the **run_distributed_task.sh** script for starting the training job.

```
# Change the set_env.sh path as needed.
source /usr/local/Ascend/ascend-toolkit/set_env.sh
mkdir -p "${output_dir}/finetune_dataset" # Retain the value.
python /home/ma-user/MA_Turbo/src/open_source/MindSpeed-LLM/preprocess_data.py \ # Location of the MindSpeed-LLM convert code pre-installed in the image. No modification is required.
    --input ${dataset_path}/alpaca_gpt4_data.json \                  # Path of the dataset to be processed.
    --tokenizer-name-or-path "$TOKENIZER_PATH" \                  # Path of the model used to process the dataset, which is set in run_distributed_task.sh.
    --output-prefix "$DATA_PATH" \                  # Path and format of the output dataset, which is set in run_distributed_task.sh.
    --handler-name AlpacaStyleInstructionHandler \    # Modify this if the format is changed.
    --tokenizer-type PretrainedFromHF \
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
    --workers 4 \
    --log-interval 1000 \
    --enable-thinking true \
    --prompt-type qwen3
```

Fine-Tuning Task Execution Script

The content of the **tune_qwen3_32b_4K_full_ptd.sh** script is as follows. For details about the parameters to be modified in the script, see the comments in the script.

```
#!/bin/bash
export HCCL_CONNECT_TIMEOUT=7200   # Maximum timeout interval. You can change the value as needed.
export HCCL_EXEC_TIMEOUT=7200
export CUDA_DEVICE_MAX_CONNECTIONS=1
export PYTORCH_NPU_ALLOC_CONF=expandable_segments:True
# ==========================================================
# 1. Receive external environment variables. (If no external environment variables are transferred, the default values after the colons are used.)
You are advised to set the environment variables in run_distributed_task.sh.
# ==========================================================
NPUS_PER_NODE=${NPUS_PER_NODE:-8}
MASTER_ADDR=${MASTER_ADDR:-"localhost"}
MASTER_PORT=${MASTER_PORT:-6000}
NNODES=${NNODES:-1}
NODE_RANK=${NODE_RANK:-0}
WORLD_SIZE=$(($NPUS_PER_NODE*$NNODES))
# The path configuration is also changed to receive environment variables. You are advised to set the environment variables in run_distributed_task.sh.
CKPT_LOAD_DIR=${CKPT_LOAD_DIR:-"${output_dir}/model_weights/qwen3_mcore/"}    # Default reference value, which is overwritten in run_distributed_task.sh
CKPT_SAVE_DIR=${CKPT_SAVE_DIR:-"${output_dir}/ckpt/qwen3-32b/"}               # Default reference value, which is overwritten in run_distributed_task.sh
DATA_PATH=${DATA_PATH:-"${output_dir}/finetune_dataset/alpaca"}               # Default reference value, which is overwritten in run_distributed_task.sh
TOKENIZER_PATH=${TOKENIZER_PATH:-"${model_path}/Qwen3-32B/"}                  # Default reference value, which is overwritten in run_distributed_task.sh)
# ==========================================================
# 2. Training parameters
# ==========================================================
TP=8     # The value must be the same as that in ckpt_convert_qwen3_hf2mcore.sh.
PP=2     # The value must be the same as that in ckpt_convert_qwen3_hf2mcore.sh.
MBS=1
GBS=32
TRAIN_ITERS=50
SEQ_LENGTH=4096
DISTRIBUTED_ARGS="
    --nproc_per_node $NPUS_PER_NODE \
    --nnodes $NNODES \
    --node_rank $NODE_RANK \
    --master_addr $MASTER_ADDR \
    --master_port $MASTER_PORT
"
OPTIMIZE_ARGS="
    --use-flash-attn \
    --use-fused-rotary-pos-emb \
    --use-rotary-position-embeddings \
    --use-fused-swiglu \
    --use-fused-rmsnorm \
    --no-masked-softmax-fusion \
    --use-distributed-optimizer
"
TRAIN_ARGS="
    --micro-batch-size 1 \
    --global-batch-size 16 \
    --lr 1.25e-6 \
    --lr-decay-style cosine \
    --min-lr 1.25e-7 \
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
    --weight-decay 1e-1 \
    --lr-warmup-fraction 0.01 \
    --attention-dropout 0.0 \
    --init-method-std 0.01 \
    --hidden-dropout 0.0 \
    --clip-grad 1.0 \
    --adam-beta1 0.9 \
    --adam-beta2 0.95 \
    --initial-loss-scale 4096 \
    --seed 42 \
    --bf16 \
    --train-iters ${TRAIN_ITERS} \
    --seq-length ${SEQ_LENGTH} \
    --no-shared-storage
"
MODEL_PARALLEL_ARGS="
    --tensor-model-parallel-size ${TP} \
    --pipeline-model-parallel-size ${PP}
"
GPT_ARGS="
    --use-mcore-models \
    --spec mindspeed_llm.tasks.models.spec.qwen3_spec layer_spec \
    --kv-channels 128 \
    --qk-layernorm \
    --tokenizer-name-or-path ${TOKENIZER_PATH} \
    --max-position-embeddings ${SEQ_LENGTH} \
    --num-layers 64 \
    --hidden-size 5120 \
    --ffn-hidden-size 25600 \
    --num-attention-heads 64 \
    --tokenizer-type PretrainedFromHF \
    --make-vocab-size-divisible-by 1 \
    --padded-vocab-size 151936 \
    --rotary-base 1000000 \
    --untie-embeddings-and-output-weights \
    --disable-bias-linear \
    --position-embedding-type rope \
    --normalization RMSNorm \
    --swiglu \
    --attention-softmax-in-fp32 \
    --no-gradient-accumulation-fusion \
    --group-query-attention \
    --num-query-groups 8
"
DATA_ARGS="
    --data-path $DATA_PATH \
    --split 100,0,0
"
OUTPUT_ARGS="
    --load ${CKPT_LOAD_DIR} \
    --save ${CKPT_SAVE_DIR} \
    --log-interval 1 \
    --save-interval ${TRAIN_ITERS} \
    --eval-interval ${TRAIN_ITERS} \
    --eval-iters 0 \
    --no-load-optim \
    --no-load-rng
"
TUNE_ARGS="
    --finetune \
    --stage sft \
    --is-instruction-dataset \
    --tokenizer-not-use-fast \
    --prompt-type qwen3 \
    --variable-seq-lengths
"
# Run torchrun. Note that NODE_RANK is added to the log file name to prevent conflicts.
mkdir -p "$CHECK_LOG_DIR" # Verification log, which does not need to be modified
set -o pipefail           # Enable pipefail.
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
torchrun $DISTRIBUTED_ARGS /home/ma-user/MA_Turbo/src/open_source/MindSpeed-LLM/
posttrain_gpt.py \  # Location of the MindSpeed-LLM startup code pre-installed in the image. No
modification is required.
    $GPT_ARGS \
    $DATA_ARGS \
    $OUTPUT_ARGS \
    $OPTIMIZE_ARGS \
    $TRAIN_ARGS \
    $TUNE_ARGS \
    $MODEL_PARALLEL_ARGS \
    --distributed-backend nccl \
    2>&1 | tee "$CHECK_LOG_PATH" && \
    echo "Training completes" >> "$CHECK_LOG_PATH"  # Append a completion flag to the log. This
environment variable is centrally configured in run_distributed_task.sh and does not require manual
modification.

sleep 10s
tail -n 500 "$CHECK_LOG_PATH" | grep -q -E "Training completes" || exit 1 # Verifies the abnormal status.
```

Weight Conversion Script (mg2hf)

The content of the **ckpt_convert_qwen3_mcore2hf.sh** script is as follows. For details about the parameters to be modified in the script, see the comments in the script.

The environment variables are set in the **run_distributed_task.sh** script for starting the training job.

The directory for storing the converted weight **only contains the model weight file**. The **config.json** model configuration file and vocabulary files such as **tokenizer.model** and **vocab.json** are not generated. The **run_distributed_task.sh** script copies the **config.json** model configuration file and vocabulary files such as **tokenizer.model** and **vocab.json** to the converted model folder.

The **convert_ckpt.py** script is used for hf2mg format conversion. Here, the **convert_ckpt.py** script is also used for mg2hf.

The parameters are described in the comments below. **When using the script, remove the comments and all spaces after the backslash (\).**

```
# Change the ascend-toolkit path.
source /usr/local/Ascend/ascend-toolkit/set_env.sh
export CUDA_DEVICE_MAX_CONNECTIONS=1
mkdir -p "$CHECK_LOG_DIR"
set -o pipefail
python /home/ma-user/MA_Turbo/src/open_source/MindSpeed-LLM/convert_ckpt.py \ # Location of the
MindSpeed-LLM convert code pre-installed in the image. No modification is required.
    --use-mcore-models \
    --model-type GPT \
    --load-model-type mg \
    --save-model-type hf \
    --target-tensor-parallel-size 1 \ # Always set to 1, no modification needed
    --target-pipeline-parallel-size 1 \# Always set to 1, no modification needed
    --spec mindspeed_llm.tasks.models.spec.qwen3_spec layer_spec \
    --load-dir "$CKPT_SAVE_DIR" \ # The checkpoint storage directory upon training completion; matches
CKPT_SAVE_DIR in run_distributed_task.sh
    --save-dir "$TOKENIZER_PATH" \ # The path to the original model in HF format; matches
TOKENIZER_PATH in run_distributed_task.sh
    --params-dtype bf16 \
    --model-type-hf qwen3 \
    2>&1 | tee "$CHECK_LOG_PATH" && \
    echo "Convert mg2hf completes" >> "$CHECK_LOG_PATH" # Adds the completion flag to the log.
sleep 10s
tail -n 500 "$CHECK_LOG_PATH" | grep -q -E "Convert mg2hf completes" || exit 1 # Verifies the abnormal
status. Ensure that the log names are the same.
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Training Job Startup Script

The content of the **run_distributed_task.sh** script is as follows. For details about the parameters to be modified in the script, see the comments in the script.

```
#!/bin/bash
set -e
# ========================
# 1. Basic environment and network configuration
# ========================
export HCCL_CONNECT_TIMEOUT=7200   # Maximum timeout interval. You can change the value as needed.
export HCCL_EXEC_TIMEOUT=7200
export HCCL_IF_BASE_PORT=61000
export HCCL_NPU_SOCKET_PORT_RANGE="61000-61050"
unset https_proxy http_proxy proxy ASCEND_RT_VISIBLE_DEVICES
source /usr/local/Ascend/ascend-toolkit/set_env.sh
source /usr/local/Ascend/nnal/atb/set_env.sh
# Automatically obtain the local IP address.
get_current_ip() {
    local ip
    ip=$(ip -4 addr | grep -v '127.0.0.1' | grep -oP '(?<=inet\s)\d+(\.\d+){3}' | head -n1 2>/dev/null)
    if [[ -z "$ip" ]]; then
        ip=$(ip -4 addr | grep -v '127.0.0.1' | awk '/inet/ {gsub(/\/.*/,""); print $2}' | head -n1 2>/dev/null)
    fi
    echo "${ip:-}"
}
CURRENT_IP=$(get_current_ip)
# Parse ModelArts environment variables to automatically obtain the number of NPUs and nodes.
export NNODES=${VC_WORKER_NUM:-1}
export NPUS_PER_NODE=${MA_NUM_GPUS:-8}
export MASTER_PORT=6000
# Automatically calculate NODE_RANK and MASTER_ADDR.
if [ "$NNODES" -gt 1 ]; then
    MASTER_ADDR_RAW="${VC_WORKER_HOSTS%%,*}"
    export MASTER_ADDR=$(python3 -c "import socket; print(socket.gethostbyname('${MASTER_ADDR_RAW}'))")
    export NODE_RANK=$(python3 -c "import os, socket; hosts = os.environ.get('VC_WORKER_HOSTS', '').split(','); cur='${CURRENT_IP}'; print(next((i for i, h in enumerate(hosts) if socket.gethostbyname(h.strip()) == cur), 0))")
    echo "[INFO] Multi-Node: Master=$MASTER_ADDR, Rank=$NODE_RANK, Total=$NNODES"
else
    export MASTER_ADDR="localhost"
    export NODE_RANK=0
    echo "[INFO] Single-Node Mode"
fi
# ========================
# 2. Define the global path (for subscripts, which will overwrite the corresponding parameters in the fine-tuning script).
# ========================
export CKPT_LOAD_DIR="${output_dir}/model_weights/qwen3_mcore/" # Path of the converted mg format
export CKPT_SAVE_DIR="${output_dir}/ckpt/qwen3-32b/"              # CKPT save path
export DATA_PATH="${output_dir}/finetune_dataset/alpaca"         # Path and format of the processed fine-tuning dataset (alpaca in this example). The path does not end with a slash (/).
export TOKENIZER_PATH="${model_path}/Qwen3-32B/"                 # Model path
export TRAINED_HF_MODELS="${output_dir}/trained_hf_models/"      # Path of the model converted from the mg format to the hf format after training is complete.

export CHECK_LOG_DIR="/home/ma-user/logs"                        # Folder where the verification log is stored in the container. The path in the container is not the mount path and does not need to be modified. The path does not end with a slash (/).
export CHECK_LOG_FILE="tune_qwen3_32b_full_ptd.log"              # Validation log name in the container. You can set the name based on the model and task.
export CHECK_LOG_PATH="${CHECK_LOG_DIR}/${CHECK_LOG_FILE}"       # Full path to the validation log within the container. This is an internal container file and does not require manual modification.

# Ensure that the directory exists.
mkdir -p "$CKPT_SAVE_DIR"
mkdir -p "$CHECK_LOG_DIR" # Validation log.
# ========================
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
# 3. Weight conversion (hf2mg) and data processing
# ========================
echo " Starting Checkpoint & Data Conversion..."
# Perform weight conversion.
echo " Running ckpt_convert_qwen3_hf2mcore.sh..."
bash ckpt_convert_qwen3_hf2mcore.sh                  # Weight conversion script of the model.
# Perform data processing.
echo " Running data_convert_qwen3_instruction.sh..."
bash data_convert_qwen3_instruction.sh               # Data processing script of the model.
echo " Preparation Done. Creating flag file."


# ========================
# 4. Start distributed training (on all nodes).
# ========================

# Check for the existence of r+ mode (required for OBS mounting)
FILE_PATH="/home/ma-user/MA_Turbo/src/open_source/MindSpeed-LLM/mindspeed_llm/tasks/preprocess/
decoder_packed_mtf_dataset.py"
if grep -q "mmap_mode='r+'" "$FILE_PATH"; then
    # Precisely replace only the lines that contain mmap_mode='r+'.
    sed -i.bak "s/\(mmap_mode=['\"]\)\(r+\)\(['\"]\)/\1r\3/g" "$FILE_PATH"
else
    echo "mmap_mode='r+' not detected; operation is skipped."
fi

echo " Starting Training Script on Rank $NODE_RANK..."
# Run the modified training script.
bash tune_qwen3_32b_4K_full_ptd.sh               # Fine-tuning script of the model


# ========================
# 5. Weight conversion (mg2hf) and saving the model
# ========================

# Execute weight conversion (mg2hf) on the master node only.
if [ "$NODE_RANK" -eq 0 ]; then
    echo "[Master] Master. Doing Convert ckpt from mcore to hf models"
    mkdir -p "$TRAINED_HF_MODELS"
    find "$TOKENIZER_PATH" -type f ! -name "*.safetensors" -exec cp {} "$TRAINED_HF_MODELS" \;  #
Copies the original model configuration files (e.g., config.json) and tokenizer files (e.g., tokenizer.model,
vocab.json)
    bash ckpt_convert_qwen3_mcore2hf.sh                                                    # Executes the mg2hf
weight conversion script.
    mv "$TOKENIZER_PATH/mg2hf/"* "$TRAINED_HF_MODELS/"                                     # Moves the
converted weights to the corresponding directory in OBS.
    echo "All steps completes" >> "$CHECK_LOG_PATH"
else
    echo "[Worker $NODE_RANK] Task finished. Waiting for Master to complete conversion..."
    echo "All steps completes" >> "$CHECK_LOG_PATH"
fi

sleep 10s
tail -n 500 "$CHECK_LOG_PATH" | grep -q -E "All steps completes"  && exit 0 || exit 1 # Reports abnormal
status back to the platform; no modifications to environment variables are needed.
```

Place the script above in the corresponding path within your own OBS bucket.

3.1.5 Executing a Training Job

This section describes how to configure and execute a training job on the ModelArts console.

Prerequisites

- You have configured permissions on the ModelArts console. For details, see **Configuring Agency Authorization for ModelArts with One Click**. If you

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

have not configured permissions, a message will be displayed on the top of the training job page, prompting you to configure permissions.

- Upload the model weight, training data, and training scripts to an OBS bucket, for example, **obs://mindspeed/mindspeed-a2**.

Step 1: Configuring Basic Settings

1. Log in to the **ModelArts console** and choose **Model Build** > **Training**.
2. Click **Create Training Job**. Set the basic parameters of the training job according to the following table.

Table 3-6 Basic configuration

Parameter	Mandatory	Description	Example
Training Mode	Yes	The training mode can be **Fine-Tuning** or **Custom Job**. In this example, select **Custom Job**.	**Custom Job**
Name	Yes	(Mandatory) Name of the training job. The system automatically generates a name, which you can then rename.	Automatically generated
Description (Optional)	No	Job description, which helps you learn about the job information in the training job list.	**MindSpeed-LLM training**

Step 2: Defining Training Configuration

Set the key parameters of the training job according to **Table 3-7**.

Table 3-7 Training configuration

Parameter	Mandatory	Description	Example
Select Image	Yes	Select the container image required for training. In this example, select a preset image. The image environment required for MindSpeed-LLM framework training is preset on the platform and can be directly used.	**Preset Images**

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Parameter	Mandatory	Description	Example
Image address	Yes	Image path. The MindSpeed-LLM image contains the software required for training, such as the CANN package, driver, PyTorch, and MindSpeed-LLM framework code package.	**MindSpeed-LLM**

Boot Command	Yes	Boot command of the training image.
		You need to upload the training job startup script and associated script files to your OBS bucket in advance.
		Once all parameters in the scripts are configured, simply execute **run_distributed_task.sh**. This script encompasses configurations for weight conversion, data processing, fine-tuning tasks, as well as model saving and checks.
		For the boot command, you must first **cd** into the local code directory inside the container where your main training startup script is stored.
			Multi-node training
			`cd /home/ma-user/MA_Turbo/src/open_source/MindSpeed-LLM/code`
			`bash run_distributed_task.sh`
			The same script is used for single-node training.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Parameter	Mandatory	Description	Example
Code Directory	Yes	If the code is stored in OBS, select the OBS bucket path **obs://mindspeed-llm/mindspeed-llm-a2/***code_directory*. During training, all files from the OBS path are copied to the training container's directory. Any changes made to this directory are not synced back to the OBS bucket. If the files in the OBS bucket are too big, downloading them to the training container can take a long time. This makes the job start slowly. It is recommended that the script, model folder, and dataset folder of a model training job be mounted to the same OBS bucket path and used only for this training job.	**obs://mindspeed-llm/mindspeed-llm-a2/**

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Parameter	Mandatory	Description	Example
Local Code Directory	Yes	Local directory of the training container. When the training starts, the system downloads the code directory from OBS to this directory and **overwrites** the existing content. Use a **newly created directory**. MindSpeed-LLM and its dependency installation directory are in **/home/ma-user/MA_Turbo/src/open_source/MindSpeed-LLM**. Create a directory as the local code directory, for example, **/home/ma-user/MA_Turbo/src/open_source/MindSpeed-LLM/code**. If the OBS code directory is **obs://mindspeed-llm/input/**, the directory structure in the container is as follows: **/home/ma-user/MA_Turbo/src/open_source/MindSpeed-LLM/code/input/**	**/home/ma-user/MA_Turbo/src/open_source/MindSpeed-LLM/code**

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Parameter	Mandatory	Description	Example
Environment Variable	No	When using a training script, add an output environment variable. For example, set **output_dir** as the parameter name (customizable). The output path **/home/ma-user/modelarts/outputs/output_dir_0** saves preprocessed data, checkpoints, and training logs to OBS. Similarly, configure environment variables for the model path for training (e.g., **model_path**) and the dataset path to be processed (e.g., **dataset_path**). Note: To use an extended OBS path, create a folder with the same name as your OBS **storage location** in the **cloud mount path** and then mount this folder. For example, if your selected extended storage OBS path is **/test/mindspeed-llm/output_dir_0** and your cloud mount path is set to **/home/ma-user/modelarts/outputs**, setting the environment variable **output_dir** to **/home/ma-user/modelarts/outputs/output_dir_0** will ensure all outputs are correctly saved directly to OBS.	Parameter: **output_dir** Value: **/home/ma-user/modelarts/outputs/output_dir_0**

Step 3: Configuring Resources

Set the resource configuration parameters of the training job according to **Table 3-8**.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Table 3-8 Resource configuration

Parameter	Mandatory	Description	Example
Source of resources	Yes	In this example, use a dedicated resource pool. If you select a public resource pool, training process files cannot be saved.	**Dedicated resource pool**
Resource Pool	Yes	Click **Select Resource Pool**. In the dialog box displayed on the right, select a physical resource pool with an 8 x Snt9b2 specification. The resource pool must be created in advance. Hover over **View** in the **Resource Fragment** column to check fragment details and check whether the resource pool meets the training requirements.	**Physical** 8 x Snt9b2
Specifications	Yes	Select an 8 x Snt9b2 specification.	8 x Snt9b2
Compute Nodes	Yes	Qwen3-8B model training requires 8-PU resources per node. Set this parameter to **1**. Qwen3-30B-A3B model training requires 16-PU resources. Set this parameter to **2**. Qwen3-32B model training requires 16-PU resources. Set this parameter to **2**.	Qwen3-8B: **1** Qwen3-30B-A3B: **2** Qwen3-32B: **2**
Storage Mounting	Yes	Click **Add Extended Storage (OBS)** to store training artifacts.	**Extended Storage (OBS)**
		Directory: Select the training output path in the OBS bucket.	**obs://mindspeed-llm/mindspeed-llm-a2/{output}**
		Mount Path: Output path of the training artifacts in the training container.	**/home/ma-user/modelarts/outputs/output_dir_0**
		readonly: The mounted OBS path can only be read and cannot be written. The training artifacts need to be written to OBS. Do not select this option.	Do not select it.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Parameter	Mandatory	Description	Example
Job Scheduling Priority	No	Scheduling priority of the training job.	Retain the default settings.
Preemption	No	When enabled, jobs that allow preemption may be terminated and re-queued if resource pool capacity is insufficient. To avoid losing training progress, configure resumable training before enabling this function.	Do not select it.

Step 4: Configuring HA

Set the HA configuration parameters of the training job according to **Table 3-9**.

Table 3-9 HA configuration

Parameter	Mandatory	Description	Example
Maximum Restarts	No	The system marks the job as failed if it continues to fail after reaching the maximum restart limit. The default value is **3**.	Retain the default settings.
Unconditional Auto Restart	No	As long as a training exception is detected, ModelArts unconditionally restarts the training job. To prevent invalid restarts, it supports a maximum of three consecutive unconditional restarts.	Select it.
Restart Upon Suspension	No	ModelArts continuously monitors job processes to detect suspension and optimize resource usage. When this feature is enabled, suspended jobs can be automatically restarted at the process level. To prevent unnecessary restarts, ModelArts limits consecutive restarts to three.	Select it.

Step 5: Managing Access Configuration

In this example, retain the default settings for parameters related to online debugging and remote SSH in the access configuration.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Step 6: Enabling Observability

In this example, retain the default settings for **Interconnect Metrics with AOM** in the observability configuration.

Step 7: Adjusting Additional Configurations

Set the additional configuration parameters of the training job according to **Table 3-10**.

Table 3-10 Additional configuration

Parameter	Mandatory	Description	Example
Persistent Log Saving	Yes	Logs will be deleted after 30 days. If you enable this function, logs will be saved to a specified OBS path. You can also download all logs on the job details page to a local path.	Select it.
Job Visibility	No	By default, the job is visible to all users in the same workspace. **Creator**: The created training job is invisible to other users.	**Workspace**
Auto Stop	No	If this function is enabled, the job tracks its runtime. It stops automatically if the runtime surpasses the set time limit.	Do not select it.
Event Notification	No	If enabled, you will receive SMS or email notifications when specific events occur (such as job status changes or suspected suspension). These notifications may incur minor charges.	Do not select it.
Tags	No	To use the same tag for multiple cloud resources, set the tag in TMS.	Do not select it.

At the bottom of the **Create Training Job** page, click **Submit** to create the training job.

A training job runs for a period of time. You can go to the training job list or training details page to view the basic information about the training job.

After the training is complete, view the training logs and performance by referring to **Viewing the Training Output**. The training output is stored in the OBS bucket.

For more information about ModelArts training, see **Model Training**.

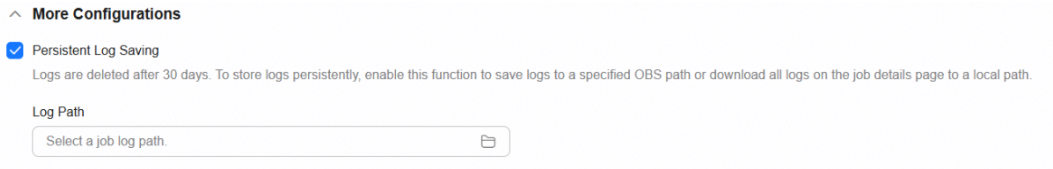
 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

3.1.6 Viewing the Training Output

Logs and Weights

The output is saved in the path you set. If you choose the output variable **${output_dir}**, you can see the output in the path configured in OBS.

Viewing Logs



You can view the logs in your OBS bucket or on the terminal of the task page.

Checking Performance

Training performance is mainly checked based on two metrics in the training logs: throughput and convergence.

You can run **npu-smi info** in the shell to view the real-time usage.

Alternatively, you can view the overall throughput and processing time in the training logs.

The logs below show each step takes about 4,500 ms and processes 4,096 tokens. Parameters such as the number of samples used in each step and the learning rate are also displayed in the logs.

Each worker's tasks can vary, and their log prints might differ. Not all workers show the training details for each step. If the primary worker does not display these details, switch to another worker.

```
[Rank 15] (after 1 iterations) memory (MB) | allocated: 35163.4208984375 | max allocated: 35179.42919921875 | reserved: 35914.0 | max reserved: 35914.0
 [2026-01-27 20:32:51] iteration        2/      50 | consumed samples:           32 | elapsed time per iteration (ms): 4553.9 | learning rate: 1.247453E-06 | global batch size:    16 | lm loss: 2.793206E+00 | loss scale: 1.0 | grad norm: 30.954 | num zeros: 0 | number of skipped iterations:   0 | number of nan iterations:   0 |
 [2026-01-27 20:32:56] iteration        3/      50 | consumed samples:           48 | elapsed time per iteration (ms): 4680.3 | learning rate: 1.242934E-06 | global batch size:    16 | lm loss: 1.886961E+00 | loss scale: 1.0 | grad norm: 20.802 | num zeros: 0 | number of skipped iterations:   0 | number of nan iterations:   0 |
 [2026-01-27 20:33:00] iteration        4/      50 | consumed samples:           64 | elapsed time per iteration (ms): 4406.0 | learning rate: 1.236179E-06 | global batch size:    16 | lm loss: 1.786803E+00 | loss scale: 1.0 | grad norm: 19.852 | num zeros: 0 | number of skipped iterations:   0 | number of nan iterations:   0 |
 [2026-01-27 20:33:05] iteration        5/      50 | consumed samples:           80 | elapsed time per iteration (ms): 4326.3 | learning rate: 1.227215E-06 | global batch size:    16 | lm loss: 1.827883E+00 | loss scale: 1.0 | grad norm: 22.299 | num zeros: 0 | number of skipped iterations:   0 | number of nan iterations:   0 |
 [2026-01-27 20:33:09] iteration        6/      50 | consumed samples:           96 | elapsed time per iteration (ms): 4609.5 | learning rate: 1.216077E-06 | global batch size:    16 | lm loss: 1.543720E+00 | loss scale: 1.0 | grad norm: 15.454 | num zeros: 0 | number of skipped iterations:   0 | number of nan iterations:   0 |
 [2026-01-27 20:33:14] iteration        7/      50 | consumed samples:          112 | elapsed time per iteration (ms): 4496.9 | learning rate: 1.202811E-06 | global batch size:    16 | lm loss: 1.908836E+00 | loss scale: 1.0 | grad norm: 24.916 | num zeros: 0 | number of skipped iterations:   0 | number of nan iterations:   0 |
 [2026-01-27 20:33:18] iteration        8/      50 | consumed samples:          128 | elapsed time per iteration (ms): 4609.6 | learning rate: 1.187470E-06 | global batch size:    16 | lm loss: 1.422310E+00 | loss scale: 1.0 | grad norm: 15.164 | num zeros: 0 | number of skipped iterations:   0 | number of nan iterations:   0 |
 [2026-01-27 20:33:23] iteration        9/      50 | consumed samples:          144 | elapsed time per iteration (ms): 4534.9 | learning rate: 1.170116E-06 | global batch size:    16 | lm loss: 1.770321E+00 | loss scale: 1.0 | grad norm: 24.568 | num zeros: 0 | number of skipped iterations:   0 | number of nan iterations:   0 |
 [2026-01-27 20:33:27] iteration       10/      50 | consumed samples:          160 | elapsed time per iteration (ms): 4600.2 | learning rate: 1.150818E-06 | global batch size:    16 | lm loss: 1.631615E+00 | loss scale: 1.0 | grad norm: 23.435 | num zeros: 0 | number of skipped iterations:   0 | number of nan iterations:   0 |
 [2026-01-27 20:33:32] iteration       11/      50 | consumed samples:          176 | elapsed time per iteration (ms): 4568.9 | learning rate: 1.129655E-06 | global batch size:    16 | lm loss: 1.527288E+00 | loss scale: 1.0 | grad norm: 20.260 | num zeros: 0 | number of skipped iterations:   0 | number of nan iterations:   0 |
 [2026-01-27 20:33:37] iteration       12/      50 | consumed samples:          192 | elapsed time per iteration (ms): 4536.4 | learning rate: 1.106711E-06 | global batch size:    16 | lm loss: 1.349640E+00 | loss scale: 1.0 | grad norm: 19.735 | num zeros: 0 | number of skipped iterations:   0 | number of nan iterations:   0 |
 [2026-01-27 20:33:41] iteration       13/      50 | consumed samples:          208 | elapsed time per iteration (ms): 4135.2 | learning rate: 1.082080E-06 | global batch size:    16 | lm loss: 1.404263E+00 | loss scale: 1.0 | grad norm: 18.840 | num zeros: 0 | number of skipped iterations:   0 | number of nan iterations:   0 |
 [2026-01-27 20:33:45] iteration       14/      50 | consumed samples:          224 | elapsed time per iteration (ms): 4516.1 | learning rate: 1.055859E-06 | global batch size:    16 | lm loss: 1.132176E+00 | loss scale: 1.0 | grad norm: 14.113 | num zeros: 0 | number of skipped iterations:   0 | number of nan iterations:   0 |
 [2026-01-27 20:33:49] iteration       15/      50 | consumed samples:          240 | elapsed time per iteration (ms): 4284.6 | learning rate: 1.028155E-06 | global batch size:    16 | lm loss: 1.173116E+00 | loss scale: 1.0 | grad norm: 15.272 | num zeros: 0 | number of skipped iterations:   0 | number of nan iterations:   0 |
 [2026-01-27 20:33:58] iteration       16/      50 | consumed samples:          256 | elapsed time per iteration (ms): 8291.0 | learning rate: 9.990800E-07 | global batch size:    16 | lm loss: 1.126942E+00 | loss scale: 1.0 | grad norm: 14.615 | num zeros: 0 | number of skipped iterations:   0 | number of nan iterations:   0 |
 [2026-01-27 20:34:04] iteration       17/      50 | consumed samples:          272 | elapsed time per iteration (ms): 6060.5 | learning rate: 9.687500E-07 | global batch size:    16 | lm loss: 1.208630E+00 | loss scale: 1.0 | grad norm: 16.973 | num zeros: 0 | number of skipped iterations:   0 | number of nan iterations:   0 |
 [2026-01-27 20:34:08] iteration       18/      50 | consumed samples:          288 | elapsed time per iteration (ms): 4322.5 | learning rate: 9.372875E-07 | global batch size:    16 | lm loss: 1.188542E+00 | loss scale: 1.0 | grad norm: 15.615 | num zeros: 0 | number of skipped iterations:   0 | number of nan iterations:   0 |
 [2026-01-27 20:34:12] iteration       19/      50 | consumed samples:          304 | elapsed time per iteration (ms): 4340.1 | learning rate: 9.048192E-07 | global batch size:    16 | lm loss: 1.178980E+00 | loss scale: 1.0 | grad norm: 11.433 | num zeros: 0 | number of skipped iterations:   0 | number of nan iterations:   0 |
 [2026-01-27 20:34:17] iteration       20/      50 | consumed samples:          320 | elapsed time per iteration (ms): 4278.4 | learning rate: 8.714757E-07 | global batch size:    16 | lm loss: 9.597773E-01 | loss scale: 1.0 | grad norm: 11.937 | num zeros: 0 | number of skipped iterations:   0 | number of nan iterations:   0 |
 [2026-01-27 20:34:21] iteration       21/      50 | consumed samples:          336 | elapsed time per iteration (ms): 4351.1 | learning rate: 8.373915E-07 | global batch size:    16 | lm loss: 1.022885E+00 | loss scale: 1.0 | grad norm: 8.334 | num zeros: 0 | number of skipped iterations:   0 | number of nan iterations:   0 |
 [2026-01-27 20:34:26] iteration       22/      50 | consumed samples:          352 | elapsed time per iteration (ms): 4466.6 | learning rate: 8.027038E-07 | global batch size:    16 | lm loss: 1.040938E+00 | loss scale: 1.0 | grad norm: 10.693 | num zeros: 0 | number of skipped iterations:   0 | number of nan iterations:   0 |
 [2026-01-27 20:34:30] iteration       23/      50 | consumed samples:          368 | elapsed time per iteration (ms): 4503.4 | learning rate: 7.675521E-07 | global batch size:    16 | lm loss: 9.208362E-01 | loss scale: 1.0 | grad norm: 14.267 | num zeros: 0 | number of skipped iterations:   0 | number of nan
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Viewing the Saved Results



After the training is complete, the data is transferred, and the task exits, you can view the saved results in the corresponding OBS path.

ckpt: Weights saved in mg format after training.

finetune_dataset: The processed dataset ready for fine-tuning.

model_weights: Model weights converted from hf to mg format.

trained_hf_models: The model weight folder containing weights converted back from mg to hf format based on the checkpoints. The script automatically copies the model configuration files (e.g., **config.json**) and tokenizer files (e.g., **tokenizer.model**, **vocab.json**) into this folder.

3.1.7 FAQs About Training

Q1: Why Does Running the Sample Script Copied from the Documentation Trigger a syntax error near unexpected token `$'{\r''? Error?

A: When you create a file by copying sample scripts from the documentation, the line endings of the Windows system might differ from those of Linux, causing line ending mismatch issues. You can perform format conversion inside the Linux system by running **dos2unix** *xxx*.**sh**, or by opening the file with **vim** and executing the following two commands:

```
:set ff=unix
:wq
```

Q2: Are the Script Parameters Provided in This Document Universally Applicable to All Models? What Should I Keep in Mind When Switching Models?

A: No. When using different models or conducting different types of training, you must refer to the specific script examples tailored to each model from **MindSpeed-LLM/docs/quick_start.md · Ascend/MindSpeed-LLM - AtomGit | GitCode**.

However, the overall workflow remains the same. You will execute the following scripts in sequence and modify their parameters based on the instructions in this document to start training:

- Weight conversion script (HF to MG format): **ckpt_convert_XXX_hf2mcore.sh**
- Data processing script: **data_convert_XXX_instruction.sh** (or **data_convert_XXX_pretrain.sh** for pre-training tasks)

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

- Training script: Select **tune** for fine-tuning, **pretrain** for pre-training, **full** for full-parameter fine-tuning, and **LoRA** for Low-Rank Adaptation.
- Weight conversion script (MG to HF format): **ckpt_convert_XXX_mcore2hf.sh**
- Main training task script: **run_distributed_task.sh** must also be adjusted accordingly based on the modifications made above.

Q3: Can I Execute the Model Weight Conversion, Data Format Conversion, or Training Task Scripts Individually Instead of Using the Main Training Startup Script?

A: This is not recommended. Beyond handling cluster communication, the main training task script (**run_distributed_task.sh**) also manages critical operations such as OBS path configurations and model saving. Furthermore, configuring environment variables within the main script is more standardized and convenient. Using the main startup script is fully supported and recommended even when training on a single instance.

Q4: If the Training Task Fails, but the Weight Conversion and Data Formatting Have Already Executed Successfully and Saved to Storage, Can I Skip Converting Them Again When I Restart the Training?

A: Yes. As long as the parameters and paths remain unchanged, you can simply comment out the weight conversion script (**ckpt_convert_XXX_hf2mcore.sh**) and the data formatting script (**data_convert_XXX_instruction.sh**) within the main training script (**run_distributed_task.sh**). Then, fill in the corresponding model paths, parallelism parameters, and directory paths in **run_distributed_task.sh** to resume or start training directly.

Q5: Where Is MindSpeed-LLM Installed? How Can I Find This Path If I Want to Make Modifications or Add Print Statements?

A: MindSpeed-LLM is installed under **/home/ma-user/MA_Turbo/src/open_source/MindSpeed-LLM**, and other dependencies are located under **/home/ma-user/MA_Turbo/src/open_source**.

Inside the MindSpeed-LLM directory, you will find the exact same branch content and structure as the official repository. When modifying or calling custom MindSpeed-LLM components and files (such as the **convert_ckpt.py** script), ensure your paths are correct, or explicitly use the absolute path: **/home/ma-user/MA_Turbo/src/open_source/MindSpeed-LLM/convert_ckpt.py**.

Q6: What Format Are the Weights Saved in After Training Is Complete? Why Does the Number of Files Differ from the Original Hugging Face (HF) Model Files?

A: Immediately after training concludes, the generated weights are in Megatron-Core (mcore) format, where weights are sharded across different GPUs to achieve maximum distributed training efficiency. After being processed by the mg2hf conversion script, these are converted into standard Hugging Face (HF) format weights, which is the universally supported format for mainstream inference engines (such as vLLM).

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

It is completely normal for the number of converted .safetensors files to differ from the original model due to differences in the configured sharding strategies. Although the file count changes, the model parameters remain fully intact under the mapping of the **model.safetensors.index.json** index file, allowing the model to be loaded and used normally.

Before using the model, generate a new **model.safetensors.index.json** based on the weights converted by mg2hf. This prevents issues where the index file still references the old, pre-trained model weight shards.

The v1 version has more examples and offers better stability; it is recommended to use v1 for weight conversion. If you prefer to use v2, ensure that both hf2mg and mg2hf utilize the v2 pipeline. You can review the documentation above to understand the differences between the two.

The weight conversion process for LoRA training differs significantly. Read the introduction in the v1 documentation to modify your training and weight conversion scripts accordingly.

Sample script to generate a new **model.safetensors.index.json**:

```
import os
import json
from safetensors import safe_open
# === Modify this to your actual path ===
models_dir = "$TRAINED_HF_MODELS"
# ==========================
def generate_index(dir_path):
    print(f"Scanning directory: {dir_path}")
    files = sorted([f for f in os.listdir(dir_path) if f.endswith(".safetensors")])

    if not files:
        print("Error: No .safetensors files found in the directory!")
        return
    weight_map = {}
    total_size = 0
    print(f"Found {len(files)} weight files. Rebuilding index...")

    for file_name in files:
        file_path = os.path.join(dir_path, file_name)
        try:
            # Open file to read the internal tensor keys
            with safe_open(file_path, framework="pt") as f:
                keys = f.keys()
                # Register all keys inside this file to the map
                for k in keys:
                    weight_map[k] = file_name

            # Get file size (optional, used for metadata)
            total_size += os.path.getsize(file_path)
            print(f"Processed: {file_name} ({len(keys)} parameters)")
        except Exception as e:
            print(f"Failed to read file {file_name}: {e}")
            return
    # Construct the final index dictionary
    index_dict = {
        "metadata": {
            "total_size": total_size
        },
        "weight_map": weight_map
    }
    # Save the new index.json
    save_path = os.path.join(dir_path, "model.safetensors.index.json")
    with open(save_path, "w") as f:
        json.dump(index_dict, f, indent=2)
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
    print("-" * 30)
    print(f"Success! New index file generated at: {save_path}")

if __name__ == "__main__":
    generate_index(models_dir)
```

3.2 Training the Qwen3 & Qwen2.5-VL Model Adapted to the verl Framework on ModelArts

3.2.1 Overview

Description

ModelArts training platform allows you to perform reinforcement learning (RL) on popular open-source models with the VeRL training framework.

VeRL

VeRL is a flexible, efficient, and production-ready RL training library for large language models (LLMs). It delivers exceptional flexibility and ease of use, which are primarily reflected in the following capabilities:

- **Effortless scalability for diverse RL algorithms**: The hybrid-controller programming model enables flexible representation and highly efficient execution of complex post-training dataflows. You can build advanced RL dataflows such as GRPO and PPO with just a few lines of code.
- **Seamless integration with existing LLM infrastructure via modular APIs**: By decoupling computation from data dependencies, VeRL seamlessly interfaces with mainstream LLM frameworks like FSDP, Megatron-LM, vLLM, and SGLang.
- **Flexible device mapping**: Models can be dynamically deployed across different sets of GPUs, achieving optimized resource utilization and high scalability across clusters of varying sizes.
- **Plug-and-play compatibility with mainstream Hugging Face Models**: Popular Hugging Face models can be directly integrated out of the box.

Supported LLMs

Table 3-11 Supported LLMs

Model	Training Scenario	Training Framework	Megatron	PyTorch FSDP	Minimum PUs
Qwen3-8b	RLHF	VeRL	x	√	8x Snt9b2
Qwen2.5-VL-32b-Instruct	RLHF	VeRL	x	√	16 x Snt9b2

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Model	Training Scenario	Training Framework	Megatron	PyTorch FSDP	Minimum PUs
Qwen3-30B-A3B	RLHF	VeRL	√	x	16 x Snt9b2

Reinforcement Learning from Human Feedback (RLHF) provides machines with a natural, human-centric interactive learning experience by incorporating human feedback directly into the training loop.

Resources and Files Required

Table 3-12 Resources and data files required for training

No.	Item	Mandatory	Description
1	ModelArts compute resources	Yes	You are advised to use Snt9b resources in a ModelArts dedicated resource pool. The pool must be created in advance.
2	OBS bucket	Yes	ModelArts requires you to create an OBS bucket for storing the model weight files, training data, training scripts, and training artifacts.
3	Model weight files	Yes	Model files for training. You must prepare them in advance. This document provides the download addresses.
4	Training datasets	Yes	Datasets for training. You must prepare them in advance. This document provides the download addresses.
5	Training scripts	Yes	Dataset preprocessing scripts and training job startup scripts. You must prepare them in advance. This document provides script examples, which can be directly used.
6	Training environment image	No	The training images are already set up on the platform. You do not need to prepare them. Just select an image when you create a training job. The image contains the software required for training, such as the CANN package, driver, PyTorch, and training framework code package.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Procedure

Table 3-13 Procedure description

Phase	Step	Description
Preparations	**Configure Cloud Resources**	Prepare the dedicated resource pool and OBS bucket required for training.
	Preparing Model Weights and Training Data	Prepare the model weight files, training datasets, and data processing script files, and upload them to the OBS bucket.
	Preparing the Training Scripts	Prepare the script files required for training in advance and upload them to the OBS bucket.
Training	**Executing a Training Job**	Create a training job. During the creation, select a preset training environment image, select the resource specifications required for training, and set input and output parameters.
Viewing the training results	**Viewing the Training Output**	View the training logs and performance results.
	Viewing the Training Output	Obtain the trained model from the output path set in the training job.

3.2.2 Configure Cloud Resources

The resources required for this guide are listed below. Prepare them in advance.

Resource Requirements

Table 3-14 Resources to be prepared

Item	Mandatory	Description	Reference
ModelArts compute resources	Yes	You are advised to use Snt9b resources in ModelArts dedicated resource pools. The dedicated resource pool must be created in advance. In addition, the dedicated resource pool must be able to access the external network through the VPC.	For details, see **Creating a Dedicated Resource Pool**.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Item	Mandatory	Description	Reference
OBS bucket	Yes	ModelArts requires you to create an OBS bucket for storing the model weight files, training data, training scripts, and training artifacts.	**Creating an OBS Bucket**

Creating an OBS Bucket

ModelArts requires you to create an OBS bucket for storing the model weight files, training data, training scripts, and training artifacts. For details, see **How Do I Create an OBS Bucket for ModelArts to Store Data?**

The following shows the folders in the OBS bucket. The planned directories must be the same as the file paths involved in the training script. After a training job is created and the OBS bucket is added to the code directory, all files in the folder will be copied to the training container. If the file size is too large, the copy speed will decrease. You are advised to only retain related files in the directory to avoid storing irrelevant content.

```
|——{OBS bucket}                          # Custom bucket name, for example, verl
  |——{verl-a2}                         # Custom OBS folder name, for example, verl-a2
    |——{models}                        # Custom OBS folder for storing various models, for example, models
      |——{Qwen3-8b}                    # Custom OBS folder for storing models, for example, Qwen3-8
      |——{Qwen2.5-VL-32b-Instruct}     # Custom OBS folder for storing models, for example, Qwen2.5-VL-32b-Instruct
      |——{Qwen3-30B-A3B}               # Custom OBS folder for storing models, for example, Qwen3-30B-A3B
    |——run_train_8b.sh                 # Startup script of the Qwen3-8B training job
    |——run_qwen3-8b_npu_ma.sh          # Execution script of the Qwen3-8B training job
    |——run_train_32b.sh                # Startup script of the Qwen2.5-VL-32b-Instruct training job
    |——run_qwen2_5_vl_32b_npu_ma.sh    # Execution script of the Qwen2.5-VL-32b-Instruct training job
    |——run_train_30b.sh                # Startup script of the Qwen3-30B-A3B training job
    |——run_qwen3_30b_a3b_npu_ma.sh     # Execution script of the Qwen3-30B-A3B training job
    |——{dataset}/                      # Custom OBS folder for storing various datasets, for example, dataset.
      |——{gsm8k}/                      # OBS folder for storing the training dataset GSM8K
      |——{geometry3k}/                 # OBS folder for storing the training dataset Geometry3K
    |——gsm8k.py                        # Data preprocessing script
    |——geometry3k.py                   # Data preprocessing script
    |——{ckpts}                         # Custom OBS folder for storing the training output, for example, ckpts
```

You can use tools such as **OBS Browser+** and **obsutil** to access and manage OBS buckets and upload and download files and data.

3.2.3 Preparing Model Weights and Training Data

Preparing Model Weight Files

This document uses the Qwen series as an example to describe the training process. **Table 3-15** lists the addresses for obtaining Qwen model weights.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Table 3-15 Supported LLMs and model weight sources

Model	Training Scenario	Training Framework	Open-Source Weight File Link
Qwen3-8B	RLHF	VeRL	**https://huggingface.co/Qwen/Qwen3-8B**
Qwen2.5-VL-32B-Instruct	RLHF	VeRL	**https://huggingface.co/Qwen/Qwen2.5-VL-32B-Instruct**
Qwen3-30B-A3B	RLHF	VeRL	**https://huggingface.co/Qwen/Qwen3-30B-A3B**

NOTE

To download weight files from the Hugging Face website, set up a proxy. For instructions, search online.

Upload the downloaded model weight files to your OBS bucket. The following is an example of the directory for storing files in your OBS bucket based on the OBS plan:

```
obs://verl/verl-a2/models/Qwen3-8B
obs://verl/verl-a2/models/Qwen2.5-VL-32b-Instruct
obs://verl/verl-a2/models/Qwen3-30B-A3B
```

Preparing Training Data and Data Preprocessing Scripts (Qwen3-8b/Qwen3-30B-A3B)

The Qwen3-8b and Qwen3-30B-A3B models use GSM8K data for training.

1. Download the GSM8K data from **https://huggingface.co/datasets/openai/gsm8k/tree/main**.
2. Refer to the file example provided below to prepare your data preprocessing script **gsm8k.py**. When you execute the training task, the training startup script **run_train_8b.sh** will call this **gsm8k.py** file to preprocess the data.
3. Upload both the GSM8K dataset and the **gsm8k.py** data preprocessing script to your OBS bucket. The following is an example of the file structure within the OBS bucket:

   ```
   obs://verl/verl-a2/dataset/gsm8k
   obs://verl/verl-a2/gsm8k.py
   ```

gsm8k.py is as follows:

During training, the default storage path for data inside the training container is **/home/ma-user/work/verl-a2/dataset/gsm8k**, which can be customized as needed. Note that **/home/ma-user/work** represents the code directory inside the training container and must match the local code directory configured when creating the training job. Meanwhile, **verl-a2/dataset/gsm8k** reflects the directory structure within your OBS bucket and must match your actual path.

```
import argparse
import os
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
import re
import datasets
from verl.utils.hdfs_io import copy, makedirs
def extract_solution(solution_str):
    solution = re.search("#### (\\-?[0-9\\.\\,]+)", solution_str)
    assert solution is not None
    final_solution = solution.group(0)
    final_solution = final_solution.split("#### ")[1].replace(",", "")
    return final_solution
if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument("--local_dir", default="~/data/gsm8k")
    parser.add_argument("--hdfs_dir", default=None)
    args = parser.parse_args()
    data_source = "openai/gsm8k"
    dataset = datasets.load_dataset("/home/ma-user/work/verl-a2/dataset/gsm8k", "main")  # "/
home/ma-user/work/verl-a2/dataset/gsm8k" is the path for storing datasets. You can change it as required.
    train_dataset = dataset["train"]
    test_dataset = dataset["test"]
    instruction_following = 'Let\'s think step by step and output the final answer after "####".'

    # add a row to each data item that represents a unique id
    def make_map_fn(split):
        def process_fn(example, idx):
            question_raw = example.pop("question")
            question = question_raw + " " + instruction_following
            answer_raw = example.pop("answer")
            solution = extract_solution(answer_raw)
            data = {
                "data_source": data_source,
                "prompt": [
                    {
                        "role": "user",
                        "content": question,
                    }
                ],
                "ability": "math",
                "reward_model": {"style": "rule", "ground_truth": solution},
                "extra_info": {
                    "split": split,
                    "index": idx,
                    "answer": answer_raw,
                    "question": question_raw,
                },
            }
            return data
        return process_fn
    train_dataset = train_dataset.map(function=make_map_fn("train"), with_indices=True)
    test_dataset = test_dataset.map(function=make_map_fn("test"), with_indices=True)
    local_dir = args.local_dir
    hdfs_dir = args.hdfs_dir
    train_dataset.to_parquet(os.path.join(local_dir, "train.parquet"))
    test_dataset.to_parquet(os.path.join(local_dir, "test.parquet"))
    if hdfs_dir is not None:
        makedirs(hdfs_dir)
        copy(src=local_dir, dst=hdfs_dir)
```

Preparing Training Data and Data Preprocessing Scripts (Qwen2.5-VL-32b-Instruct)

The Qwen2.5-VL-32b-Instruct model uses geometry3k data for training.

1. Download the geometry3k data from **https://huggingface.co/datasets/hiyouga/geometry3k/tree/main**.
2. Refer to the file example provided below to prepare your data preprocessing script **geometry3k.py**. When you execute the training task, the training

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

startup script **run_train_32b.sh** will call this **geometry3k.py** file to preprocess the data.

3. Upload both the **geometry3k** dataset and the **geometry3k.py** data preprocessing script to your OBS bucket. The following is an example of the file structure within the OBS bucket:

```
obs://verl/verl-a2/dataset/geometry3k
obs://verl/verl-a2/geometry3k.py
```

geometry3k.py is as follows:

During training, the default storage path for data inside the training container is **/home/ma-user/work/verl-a2/dataset/geometry3k**, which can be customized as needed. Note that **/home/ma-user/work** represents the code directory inside the training container and must match the local code directory configured when creating the training job. Meanwhile, **verl-a2/dataset/geometry3k** reflects the directory structure within your OBS bucket and must match your actual path.

```
import argparse
import os
import datasets
from verl.utils.hdfs_io import copy, makedirs
if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument("--local_dir", default="~/data/geo3k")
    parser.add_argument("--hdfs_dir", default=None)
    args = parser.parse_args()
    data_source = "hiyouga/geometry3k"
    dataset = datasets.load_dataset("/home/ma-user/work/verl-a2/dataset/geometry3k")  # "/home/ma-user/work/verl-a2/dataset/geometry3k" is the dataset path. You can change it as required.
    train_dataset = dataset["train"]
    test_dataset = dataset["test"]
    instruction_following = (
        r"You FIRST think about the reasoning process as an internal monologue and then provide the final answer. "
        r"The reasoning process MUST BE enclosed within <think> </think> tags. The final answer MUST BE put in \boxed{}."
    )
    # add a row to each data item that represents a unique id
    def make_map_fn(split):
        def process_fn(example, idx):
            problem = example.pop("problem")
            prompt = problem + " " + instruction_following
            answer = example.pop("answer")
            images = example.pop("images")
            data = {
                "data_source": data_source,
                "prompt": [
                    {
                        "role": "user",
                        "content": prompt,
                    }
                ],
                "images": images,
                "ability": "math",
                "reward_model": {"style": "rule", "ground_truth": answer},
                "extra_info": {
                    "split": split,
                    "index": idx,
                    "answer": answer,
                    "question": problem,
                },
            }
            return data
        return process_fn
    train_dataset = train_dataset.map(function=make_map_fn("train"), with_indices=True, num_proc=8)
    test_dataset = test_dataset.map(function=make_map_fn("test"), with_indices=True, num_proc=8)
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
local_dir = args.local_dir
hdfs_dir = args.hdfs_dir
train_dataset.to_parquet(os.path.join(local_dir, "train.parquet"))
test_dataset.to_parquet(os.path.join(local_dir, "test.parquet"))
if hdfs_dir is not None:
    makedirs(hdfs_dir)
    copy(src=local_dir, dst=hdfs_dir)
```

3.2.4 Preparing the Training Scripts

For details about the training script files used for model training, see **Training Scripts (Qwen3-8B)**, **Training Scripts (Qwen2.5-VL-32B-Instruct)**, and **Training Scripts (Qwen3-30B-A3B)**. You need to create an SH file by referring to this section and upload it to your OBS bucket.

```
Example of the file structure within the OBS bucket:
obs://verl/verl-a2/run_train_8b.sh
obs://verl/verl-a2/run_qwen3-8b_npu_ma.sh
obs://verl/verl-a2/run_train_32b.sh
obs://verl/verl-a2/run_qwen2_5_vl_32b_npu_ma.sh
obs://verl/verl-a2/run_train_30b.sh
obs://verl/verl-a2/run_qwen3_30b_a3b_npu_ma.sh
```

Table 3-16 Training scripts

Model	Training Script	Script Description
Qwen3-8B	run_train_8b.sh	Training job startup script, which handles data preprocessing, starting the training job, and outputting training logs.
	run_qwen3-8b_npu_ma.sh	Used to start the VeRL reinforcement learning training job.
Qwen2.5-VL-32B-Instruct	run_train_32b.sh	Training job startup script, which handles data preprocessing, starting the training job, and outputting training logs.
	run_qwen2_5_vl_32b_npu_ma.sh	Used to start the VeRL reinforcement learning training job.
Qwen3-30B-A3B	run_train_30b.sh	Training job startup script, which handles data preprocessing, starting the training job, and outputting training logs.
	run_qwen3_30b_a3b_npu_ma.sh	Used to start the VeRL reinforcement learning training job.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Path Description

The paths involved in the training script files are defined as follows:

- **/home/ma-user/work**: The local code directory configured when creating the ModelArts training job. This can be customized, but the paths specified in the scripts must perfectly match the paths set during training job creation.
- **/verl-a2**: The name of the OBS folder used to store code, datasets, and model weights. This can be customized.
- **/output_dir**: The cloud mount path configured when mounting OBS during the training job creation. It is used to save preprocessed data, checkpoints, and training logs. This can be customized.
- **/home/ma-user/verl**: The pre-installed verl code repository built into the container image. No modifications are required.

Parameter Description

Parameter	Description	Example Value
data.train_files	Training set after preprocessing	**~/dataset/gsm8k/precessed/train.parquet**
data.val_files	Validation set after preprocessing	**~/dataset/gsm8k/precessed/test.parquet**
actor_rollout_ref.model.path	Hugging Face model path, which can also be a path in the container.	**~/models/Qwen3-8B**
trainer.n_gpus_per_node	Number of NPUs per node.	**8**
trainer.nnodes	Compute nodes required for training.	**2**
algorithm.adv_estimator	Reinforcement learning policy algorithm.	**grpo**
data.train_batch_size	Sampling batch size per training iteration.	**16**
data.max_prompt_length	Maximum prompt length.	**2048**
data.max_response_length	Maximum response length.	**4096**
actor_rollout_ref.rollout.tensor_model_parallel_size	Model segmentation during vLLM inference.	**2**

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Parameter	Description	Example Value
actor_rollout_ref.nccl_timeout	Communication timeout interval. Set this parameter based on the memory access requirements.	**7200**
actor_rollout_ref.rollout.n	Number of importance sampling times.	**2**
trainer.save_freq	Checkpoint saving frequency, measured in training steps.	**50**
trainer.test_freq	Validation frequency, measured in training steps.	**50**
trainer.total_epochs	Number of training epochs. One epoch is when the model traverses the entire training set once.	**1**
trainer.total_training_steps	Number of training steps. One step is when the model completes one forward pass, one backward pass, and one parameter update.	**100**

For details about more parameters, see the **official VeRL documentation**.

Training Scripts (Qwen3-8B)

You need to prepare two training script files: **run_train_8b.sh** and **run_qwen3-8b_npu_ma.sh**.

- **run_train_8b.sh** is the training job startup script, which handles data preprocessing, starting the training task, and outputting training logs. When executed, the **run_train_8b.sh** script automatically calls the data preprocessing script **gsm8k.py** and the VeRL training script **run_qwen3-8b_npu_ma.sh**.

  The content of the **run_train_8b.sh** script is as follows:

```
#!/bin/bash
set -e
export HCCL_IF_BASE_PORT=61000
export HCCL_NPU_SOCKET_PORT_RANGE="61000-61050"
export HCCL_EXEC_TIMEOUT=60000
export HCCL_CONNECT_TIMEOUT=7200
unset https_proxy http_proxy proxy ASCEND_RT_VISIBLE_DEVICES
source /usr/local/Ascend/ascend-toolkit/set_env.sh
source /usr/local/Ascend/nnal/atb/set_env.sh
export NNODES=${VC_WORKER_NUM:-1}
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
export NGPUS_PER_NODE=${MA_NUM_GPUS:-8}
export MASTER_ADDR="${VC_WORKER_HOSTS%%,*}"
export DOMAIN_IP=$(python -c "import socket; print(socket.gethostbyname('${MASTER_ADDR}'))")
export MASTER_NODE_IP="${DOMAIN_IP}"
export MASTER_NODE_PORT="6699"
export USE_OPTIMIZED_MODEL=0
export VLLM_VERSION=0.16.0
export PYTHONPATH="/home/ma-user/Megatron-LM:/home/ma-user/vllm${PYTHONPATH:+:$
{PYTHONPATH}}"
MAX_WAIT_SECONDS=900
CHECK_INTERVAL=3
get_current_ip() {
    local ip
    ip=$(ip -4 addr | grep -v '127.0.0.1' | grep -oP '(?<=inet\s)\d+(\.\d+){3}' | head -n1 2>/dev/null)
    if [[ -z "$ip" ]]; then
        ip=$(ip -4 addr | grep -v '127.0.0.1' | awk '/inet/ {gsub(/\/.*/,""); print $2}' | head -n1 2>/dev/null)
    fi
    echo "${ip:-}"
}
CURRENT_IP=$(get_current_ip)
if [[ -z "$CURRENT_IP" ]]; then
    echo "Failed to obtain the IP address of the current node."
    exit 1
fi
echo "current ip: $CURRENT_IP, master ip: $MASTER_NODE_IP"
if [ "$CURRENT_IP" = "$MASTER_NODE_IP" ]; then
    echo "start Ray master node..."
    ray stop --force
    ray start --head --port="$MASTER_NODE_PORT"
    elapsed_seconds=0
    echo "wait $NNODES nodes..."
    while true; do
        current_node_num=$(ray status | grep -c "node_" || echo 0)
        if ! [[ "$current_node_num" =~ ^[0-9]+$ ]]; then
            current_node_num=0
        fi
        echo "current_node_num: $current_node_num"
        if [ "$current_node_num" -ge "$NNODES" ]; then
            echo "all $NNODES workers node is ready."
            break
        fi
        if [ "$elapsed_seconds" -ge "$MAX_WAIT_SECONDS" ]; then
            echo "timeout ($MAX_WAIT_SECONDS s), current_node_num: $current_node_num"
            ray stop --force
            exit 1
        fi
        sleep "$CHECK_INTERVAL"
        elapsed_seconds=$((elapsed_seconds + CHECK_INTERVAL))
    done
    echo "start job....."
    RAY_DATA_HOME=${RAY_DATA_HOME:-"/home/ma-user/work/verl-a2"}  # /home/ma-user/work
is the local code directory configured when creating the ModelArts training job. This can be
customized, but the path in the script must match the path set during training job creation. /verl-a2
is the name of the OBS folder used to store code, datasets, and model weights, which can also be
customized.
    OUTPUT_DIR=${OUTPUT_DIR:-"/output_dir/verl-a2"}               # /output_dir is the cloud mount
path configured when mounting the OBS bucket (/verl-a2) during training job creation. It is used to
save preprocessed data, checkpoints, and training logs, and can be customized.
    cd ${RAY_DATA_HOME}
    if [ ! -d "$OUTPUT_DIR/dataset/gsm8k/precessed" ]; then
        python3 gsm8k.py --local_dir "$OUTPUT_DIR/dataset/gsm8k/precessed"
    fi
    cp ${RAY_DATA_HOME}/run_qwen3-8b_npu_ma.sh /home/ma-user/verl/examples/grpo_trainer/
run_qwen3-8b_npu_ma.sh
    cd /home/ma-user/verl
    bash examples/grpo_trainer/run_qwen3-8b_npu_ma.sh
else
    echo "start workers, connect to master: $MASTER_NODE_IP:$MASTER_NODE_PORT"
    ray start \
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
        --address="${MASTER_NODE_IP}:${MASTER_NODE_PORT}"
fi
```

- The **run_qwen3-8b_npu_ma.sh** script is used to start the VeRL reinforcement learning training job. **Path Description** describes the paths requiring modification.

  The content of the **run_qwen3-8b_npu_ma.sh** script is as follows:

```
set -x
project_name='GRPO-Qwen3'
exp_name='GRPO-Qwen3-8B-npu'
gen_tp=2
RAY_DATA_HOME=${RAY_DATA_HOME:-"/home/ma-user/work/verl-a2"}                    # /home/ma-user/work is the local code directory configured when creating the ModelArts training job. This can be customized, but the path in the script must match the path set during training job creation. /verl-a2 is the name of the OBS folder used to store code, datasets, and model weights, which can be customized.
OUTPUT_DIR=${OUTPUT_DIR:-"/output_dir/verl-a2"}                                 # /output_dir is the cloud mount path configured when mounting the OBS bucket (/verl-a2) during training job creation. It is used to save preprocessed data, checkpoints, and training logs, and can be customized.
MODEL_PATH=${MODEL_PATH:-"${RAY_DATA_HOME}/models/Qwen3-8B"}                    # .../models/Qwen3-8B is the OBS folder where the model is stored, which can be customized.
CKPTS_DIR=${CKPTS_DIR:-"${OUTPUT_DIR}/ckpts/${project_name}/${exp_name}"}       # .../ckpts is the OBS folder where training outputs are saved, which can be customized.
TRAIN_FILE=${TRAIN_FILE:-"${OUTPUT_DIR}/dataset/gsm8k/precessed/train.parquet"} # .../dataset/gsm8k is the OBS folder where the GSM8K training data is stored, which can be customized. /precessed/train.parquet is the preprocessed training data and does not need to be modified.
TEST_FILE=${TEST_FILE:-"${OUTPUT_DIR}/dataset/gsm8k/precessed/test.parquet"}    # .../dataset/gsm8k is the OBS folder where the GSM8K training data is stored, which can be customized. /precessed/test.parquet is the preprocessed test data and does not need to be modified.
python3 -m verl.trainer.main_ppo \
    algorithm.adv_estimator=grpo \
    data.train_files="${TRAIN_FILE}" \
    data.val_files="${TEST_FILE}" \
    data.train_batch_size=16 \
    data.max_prompt_length=2048 \
    data.max_response_length=4096 \
    data.filter_overlong_prompts=True \
    data.truncation='error' \
    actor_rollout_ref.model.path=${MODEL_PATH} \
    actor_rollout_ref.actor.optim.lr=1e-6 \
    actor_rollout_ref.model.use_remove_padding=True \
    actor_rollout_ref.actor.ppo_mini_batch_size=8 \
    actor_rollout_ref.actor.ppo_micro_batch_size_per_gpu=1 \
    actor_rollout_ref.actor.use_kl_loss=True \
    actor_rollout_ref.actor.kl_loss_coef=0.001 \
    actor_rollout_ref.actor.kl_loss_type=low_var_kl \
    actor_rollout_ref.actor.entropy_coeff=0 \
    actor_rollout_ref.actor.use_torch_compile=False \
    actor_rollout_ref.nccl_timeout=7200 \
    actor_rollout_ref.ref.use_torch_compile=False \
    actor_rollout_ref.model.enable_gradient_checkpointing=True \
    actor_rollout_ref.actor.fsdp_config.param_offload=True \
    actor_rollout_ref.actor.fsdp_config.optimizer_offload=True \
    actor_rollout_ref.rollout.log_prob_micro_batch_size_per_gpu=1 \
    actor_rollout_ref.rollout.tensor_model_parallel_size=${gen_tp} \
    actor_rollout_ref.rollout.name=vllm \
    actor_rollout_ref.rollout.gpu_memory_utilization=0.6 \
    actor_rollout_ref.rollout.n=2 \
    actor_rollout_ref.ref.log_prob_micro_batch_size_per_gpu=1 \
    actor_rollout_ref.ref.fsdp_config.param_offload=True \
    actor_rollout_ref.rollout.checkpoint_engine.update_weights_bucket_megabytes=4096 \
    algorithm.use_kl_in_reward=False \
    trainer.critic_warmup=0 \
    trainer.logger=console \
    trainer.project_name="${project_name}" \
    trainer.experiment_name="${exp_name}" \
    trainer.n_gpus_per_node=${NGPUS_PER_NODE} \
    trainer.nnodes=${NNODES} \
    trainer.default_local_dir=${CKPTS_DIR} \
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
    trainer.resume_mode=auto \
    actor_rollout_ref.actor.fsdp_config.forward_prefetch=True \
    actor_rollout_ref.ref.fsdp_config.forward_prefetch=True \
    ++actor_rollout_ref.actor.entropy_from_logits_with_chunking=True \
    ++actor_rollout_ref.ref.entropy_from_logits_with_chunking=True \
    trainer.val_before_train=True \
    trainer.save_freq=50 \
    trainer.test_freq=50 \
    trainer.total_epochs=1 \
    trainer.total_training_steps=100 2>&1 | tee ${OUTPUT_DIR}/run_qwen3-8b_npu_ma.log

sleep 20s
if tail -n 100 ${OUTPUT_DIR}/run_qwen3-8b_npu_ma.log | grep -q 'Final validation metrics'; then
    exit 0
else
    exit 1
fi
```

Training Scripts (Qwen2.5-VL-32B-Instruct)

You need to prepare two training script files: **run_train_32b.sh** and **run_qwen3-8b_npu_ma.sh**.

- **run_train_32b.sh** is the training job startup script, which handles data preprocessing, starting the training task, and outputting training logs. When executed, the **run_train_32b.sh** script automatically calls the data preprocessing script **geometry3k.py** and the VeRL training script **run_qwen2_5_vl_32b_npu_ma.sh**.

  The content of the **run_train_32b.sh** script is as follows:

```
#!/bin/bash
set -e
export HCCL_IF_BASE_PORT=61000
export HCCL_NPU_SOCKET_PORT_RANGE="61000-61050"
export HCCL_EXEC_TIMEOUT=60000
export HCCL_CONNECT_TIMEOUT=7200
unset https_proxy http_proxy proxy ASCEND_RT_VISIBLE_DEVICES
source /usr/local/Ascend/ascend-toolkit/set_env.sh
source /usr/local/Ascend/nnal/atb/set_env.sh
# Change the values as needed.
export NNODES=${VC_WORKER_NUM:-2}
export NGPUS_PER_NODE=${MA_NUM_GPUS:-8}
export MASTER_ADDR="${VC_WORKER_HOSTS%%,*}"
export DOMAIN_IP=$(python -c "import socket; print(socket.gethostbyname('${MASTER_ADDR}'))")
export MASTER_NODE_IP="${DOMAIN_IP}"
export MASTER_NODE_PORT="6699"
export USE_OPTIMIZED_MODEL=0
export VLLM_VERSION=0.16.0
export PYTHONPATH="/home/ma-user/Megatron-LM:/home/ma-user/vllm${PYTHONPATH:+:$
{PYTHONPATH}}"
MAX_WAIT_SECONDS=900
CHECK_INTERVAL=3
get_current_ip() {
    local ip
    ip=$(ip -4 addr | grep -v '127.0.0.1' | grep -oP '(?<=inet\s)\d+(\.\d+){3}' | head -n1 2>/dev/null)
    if [[ -z "$ip" ]]; then
        ip=$(ip -4 addr | grep -v '127.0.0.1' | awk '/inet/ {gsub(/\/.*/,""); print $2}' | head -n1 2>/dev/null)
    fi
    echo "${ip:-}"
}
CURRENT_IP=$(get_current_ip)
if [[ -z "$CURRENT_IP" ]]; then
    echo "Failed to obtain the IP address of the current node."
    exit 1
fi
echo "current ip: $CURRENT_IP, master ip: $MASTER_NODE_IP"
if [ "$CURRENT_IP" = "$MASTER_NODE_IP" ]; then
    echo "start Ray master node..."
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
    ray stop --force
    ray start --head --port="$MASTER_NODE_PORT"
    elapsed_seconds=0
    echo "wait $NNODES nodes..."
    while true; do
        current_node_num=$(ray status | grep -c "node_" || echo 0)
        if ! [[ "$current_node_num" =~ ^[0-9]+$ ]]; then
            current_node_num=0
        fi
        echo "current_node_num: $current_node_num"
        if [ "$current_node_num" -ge "$NNODES" ]; then
            echo "all $NNODES workers node is ready."
            break
        fi
        if [ "$elapsed_seconds" -ge "$MAX_WAIT_SECONDS" ]; then
            echo "timeout ($MAX_WAIT_SECONDS s), current_node_num: $current_node_num"
            ray stop --force
            exit 1
        fi
        sleep "$CHECK_INTERVAL"
        elapsed_seconds=$((elapsed_seconds + CHECK_INTERVAL))
    done
    echo "start job....."
    RAY_DATA_HOME=${RAY_DATA_HOME:-"/home/ma-user/work/verl-a2"}  # /home/ma-user/work is the local code directory configured when creating the ModelArts training job. This can be customized, but the path in the script must match the path set during training job creation. /verl-a2 is the name of the OBS folder used to store code, datasets, and model weights, which can also be customized.
    OUTPUT_DIR=${OUTPUT_DIR:-"/output_dir/verl-a2"}               # /output_dir is the cloud mount path configured when mounting the OBS bucket (/verl-a2) during training job creation. It is used to save preprocessed data, checkpoints, and training logs, and can be customized.
    cd ${RAY_DATA_HOME}
    if [ ! -d "$OUTPUT_DIR/dataset/geometry3k/precessed" ]; then
        python3 geometry3k.py --local_dir "$OUTPUT_DIR/dataset/geometry3k/precessed"
    fi
    cp ${RAY_DATA_HOME}/run_qwen2_5_vl_32b_npu_ma.sh /home/ma-user/verl/examples/grpo_trainer/run_qwen2_5_vl_32b_npu_ma.sh
    cd /home/ma-user/verl
    bash examples/grpo_trainer/run_qwen2_5_vl_32b_npu_ma.sh
else
    echo "start workers, connect to master: $MASTER_NODE_IP:$MASTER_NODE_PORT"
    ray start \
        --address="${MASTER_NODE_IP}:${MASTER_NODE_PORT}"
fi
```

- The **run_qwen2_5_vl_32b_npu_ma.sh** script is used to start the VeRL reinforcement learning training job. **Path Description** describes the paths requiring modification.

  The content of the **runqwen2_5_vl_32b_npu_ma.sh** script is as follows:

```
set -x
project_name='GRPO-Qwen2_5'
exp_name='GRPO-Qwen2_5-VL-32B-npu'
RAY_DATA_HOME=${RAY_DATA_HOME:-"/home/ma-user/work/verl-a2"}                    # /home/ma-user/work is the local code directory configured when creating the ModelArts training job. This can be customized, but the path in the script must match the path set during training job creation. /verl-a2 is the name of the OBS folder used to store code, datasets, and model weights, which can be customized.
OUTPUT_DIR=${OUTPUT_DIR:-"/output_dir/verl-a2"}                                 # /output_dir is the cloud mount path configured when mounting the OBS bucket (/verl-a2) during training job creation. It is used to save preprocessed data, checkpoints, and training logs, and can be customized.
MODEL_PATH=${MODEL_PATH:-"${RAY_DATA_HOME}/models/Qwen2.5-VL-32B-Instruct"}
# .../models/Qwen2.5-VL-32B-Instruct is the OBS folder where the model is stored, which can be customized.
CKPTS_DIR=${CKPTS_DIR:-"${OUTPUT_DIR}/ckpts/${project_name}/${exp_name}"}      # .../ckpts is the OBS folder where training outputs are saved, which can be customized.
TRAIN_FILE=${TRAIN_FILE:-"${OUTPUT_DIR}/dataset/geometry3k/precessed/train.parquet"}  # .../dataset/geometry3k is the OBS folder where the Geometry3K training data is stored, which can be customized. /precessed/train.parquet is the preprocessed training data and does not need to be
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

modified.
TEST_FILE=${TEST_FILE:-"${OUTPUT_DIR}/dataset/geometry3k/precessed/test.parquet"} # **.../dataset/geometry3k** is the OBS folder where the Geometry3K training data is stored, which can be customized. **/precessed/test.parquet** is the preprocessed test data and does not need to be modified.
ENGINE=*${1*:-vllm*}*
# Some models are optimized by vllm ascend. While in some case, e.g. rlhf training,
# the optimized model may not be suitable. In this case, set this value to 0 to disable the optimized model.
export USE_OPTIMIZED_MODEL=0
python3 -m verl.trainer.main_ppo \
    algorithm.adv_estimator=grpo \
    data.train_files="${TRAIN_FILE}" \
    data.val_files="${TEST_FILE}" \
    data.train_batch_size=16 \
    data.max_prompt_length=2048 \
    data.max_response_length=4096 \
    data.filter_overlong_prompts=True \
    data.truncation='error' \
    data.image_key=images \
    actor_rollout_ref.model.path="${MODEL_PATH}" \
    actor_rollout_ref.actor.optim.lr=1e-6 \
    actor_rollout_ref.model.use_remove_padding=True \
    actor_rollout_ref.actor.ppo_mini_batch_size=16 \
    actor_rollout_ref.actor.ppo_micro_batch_size_per_gpu=1 \
    actor_rollout_ref.actor.use_kl_loss=True \
    actor_rollout_ref.actor.kl_loss_coef=0.01 \
    actor_rollout_ref.actor.kl_loss_type=low_var_kl \
    actor_rollout_ref.actor.entropy_coeff=0 \
    actor_rollout_ref.actor.use_torch_compile=False \
    actor_rollout_ref.nccl_timeout=7200 \
    actor_rollout_ref.model.enable_gradient_checkpointing=True \
    actor_rollout_ref.actor.fsdp_config.param_offload=True \
    actor_rollout_ref.actor.fsdp_config.optimizer_offload=True \
    actor_rollout_ref.rollout.log_prob_micro_batch_size_per_gpu=1 \
    actor_rollout_ref.rollout.tensor_model_parallel_size=8 \
    actor_rollout_ref.rollout.name=$ENGINE \
    actor_rollout_ref.rollout.gpu_memory_utilization=0.6 \
    actor_rollout_ref.rollout.enable_chunked_prefill=False \
    actor_rollout_ref.rollout.enforce_eager=True \
    actor_rollout_ref.rollout.free_cache_engine=True \
    actor_rollout_ref.rollout.n=2 \
    actor_rollout_ref.ref.log_prob_micro_batch_size_per_gpu=1 \
    actor_rollout_ref.ref.fsdp_config.param_offload=True \
    actor_rollout_ref.rollout.checkpoint_engine.update_weights_bucket_megabytes=4096 \
    algorithm.use_kl_in_reward=False \
    trainer.critic_warmup=0 \
    trainer.logger=console \
    trainer.project_name="${project_name}" \
    trainer.experiment_name="${exp_name}" \
    trainer.n_gpus_per_node=${NGPUS_PER_NODE} \
    trainer.nnodes=${NNODES} \
    trainer.default_local_dir=${CKPTS_DIR} \
    trainer.save_freq=50 \
    trainer.test_freq=50 \
    trainer.total_epochs=1 \
    trainer.total_training_steps=100
 2>&1 | tee ${OUTPUT_DIR}/run_qwen2_5_vl_32b_npu_ma.log

sleep 20s
if tail -n 100 ${OUTPUT_DIR}/run_qwen2_5_vl_32b_npu_ma.log | grep -q 'Final validation metrics';
then
    exit 0
else
    exit 1
fi

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Training Scripts (Qwen3-30B-A3B)

You need to prepare two training script files: **run_train_30b.sh** and **run_qwen3_30b_a3b_npu_ma.sh**.

- **run_train_30b.sh** is the training job startup script, which handles data preprocessing, starting the training task, and outputting training logs. When executed, the **run_train_30b.sh** script automatically calls the data preprocessing script **gsm8k.py** and the VeRL training script **run_qwen3_30b_a3b_npu_ma.sh**.

  The content of the **run_train_30b.sh** script is as follows:

```
#!/bin/bash
set -e
export HCCL_IF_BASE_PORT=61000
export HCCL_NPU_SOCKET_PORT_RANGE="61000-61050"
export HCCL_EXEC_TIMEOUT=60000
export HCCL_CONNECT_TIMEOUT=7200
unset https_proxy http_proxy proxy ASCEND_RT_VISIBLE_DEVICES
source /usr/local/Ascend/ascend-toolkit/set_env.sh
source /usr/local/Ascend/nnal/atb/set_env.sh
export NNODES=${VC_WORKER_NUM:-1}
export NGPUS_PER_NODE=${MA_NUM_GPUS:-8}
export MASTER_ADDR="${VC_WORKER_HOSTS%%,*}"
export DOMAIN_IP=$(python -c "import socket; print(socket.gethostbyname('${MASTER_ADDR}'))")
export MASTER_NODE_IP="${DOMAIN_IP}"
export MASTER_NODE_PORT="6699"
export USE_OPTIMIZED_MODEL=0
export VLLM_VERSION=0.16.0
export PYTHONPATH="/home/ma-user/Megatron-LM:/home/ma-user/vllm${PYTHONPATH:+:$
{PYTHONPATH}}"
MAX_WAIT_SECONDS=900
CHECK_INTERVAL=3
get_current_ip() {
    local ip
    ip=$(ip -4 addr | grep -v '127.0.0.1' | grep -oP '(?<=inet\s)\d+(\.\d+){3}' | head -n1 2>/dev/null)
    if [[ -z "$ip" ]]; then
        ip=$(ip -4 addr | grep -v '127.0.0.1' | awk '/inet/ {gsub(/\/.*/,""); print $2}' | head -n1 2>/dev/null)
    fi
    echo "${ip:-}"
}
CURRENT_IP=$(get_current_ip)
if [[ -z "$CURRENT_IP" ]]; then
    echo "Failed to obtain the IP address of the current node."
    exit 1
fi
echo "current ip: $CURRENT_IP, master ip: $MASTER_NODE_IP"
if [ "$CURRENT_IP" = "$MASTER_NODE_IP" ]; then
    echo "start Ray master node..."
    ray stop --force
    ray start --head --port="$MASTER_NODE_PORT"
    elapsed_seconds=0
    echo "wait $NNODES nodes..."
    while true; do
        current_node_num=$(ray status | grep -c "node_" || echo 0)
        if ! [[ "$current_node_num" =~ ^[0-9]+$ ]]; then
            current_node_num=0
        fi
        echo "current_node_num: $current_node_num"
        if [ "$current_node_num" -ge "$NNODES" ]; then
            echo "all $NNODES workers node is ready."
            break
        fi
        if [ "$elapsed_seconds" -ge "$MAX_WAIT_SECONDS" ]; then
            echo "timeout ($MAX_WAIT_SECONDS s), current_node_num: $current_node_num"
            ray stop --force
            exit 1
        fi
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
        sleep "$CHECK_INTERVAL"
        elapsed_seconds=$((elapsed_seconds + CHECK_INTERVAL))
    done
    echo "start job....."
    RAY_DATA_HOME=${RAY_DATA_HOME:-"/home/ma-user/work/verl-a2"}  # /home/ma-user/work is the local code directory configured when creating the ModelArts training job. This can be customized, but the path in the script must match the path set during training job creation. /verl-a2 is the name of the OBS folder used to store code, datasets, and model weights, which can also be customized.
    OUTPUT_DIR=${OUTPUT_DIR:-"/output_dir/verl-a2"}               # /output_dir is the cloud mount path configured when mounting the OBS bucket (/verl-a2) during training job creation. It is used to save preprocessed data, checkpoints, and training logs, and can be customized.
    cd ${RAY_DATA_HOME}
    if [ ! -d "$OUTPUT_DIR/dataset/gsm8k/precessed" ]; then
        python3 gsm8k.py --local_dir "$OUTPUT_DIR/dataset/gsm8k/precessed"
    fi
    cp ${RAY_DATA_HOME}/run_qwen3_30b_a3b_npu_ma.sh /home/ma-user/verl/examples/grpo_trainer/run_qwen3_30b_a3b_npu_ma.sh
    cd /home/ma-user/verl
    cp verl/trainer/config/ppo_megatron_trainer.yaml .
    bash examples/grpo_trainer/run_qwen3_30b_a3b_npu_ma.sh
else
    echo "start workers, connect to master: $MASTER_NODE_IP:$MASTER_NODE_PORT"
    ray start \
        --address="${MASTER_NODE_IP}:${MASTER_NODE_PORT}"
fi
```

- The **run_qwen3_30b_a3b_npu_ma.sh** script is used to start the VeRL reinforcement learning training job. **Path Description** describes the paths requiring modification.

  The content of the **run_qwen3_30b_a3b_npu_ma.sh** script is as follows:

```
set -x
project_name='GRPO-Qwen3'
exp_name='GRPO-Qwen3-30B-A3B-npu'
offload=True
train_tp=4
train_pp=2
gen_tp=4
RAY_DATA_HOME=${RAY_DATA_HOME:-"/home/ma-user/work/verl-a2"}                  # /home/ma-user/work is the local code directory configured when creating the ModelArts training job. This can be customized, but the path in the script must match the path set during training job creation. /verl-a2 is the name of the OBS folder used to store code, datasets, and model weights, which can be customized.
OUTPUT_DIR=${OUTPUT_DIR:-"/output_dir/verl-a2"}                               # /output_dir is the cloud mount path configured when mounting the OBS bucket (/verl-a2) during training job creation. It is used to save preprocessed data, checkpoints, and training logs, and can be customized.
MODEL_PATH=${MODEL_PATH:-"${RAY_DATA_HOME}/models/Qwen3-30B-A3B"}             # .../models/Qwen3-8B is the OBS folder where the model is stored, which can be customized.
CKPTS_DIR=${CKPTS_DIR:-"${OUTPUT_DIR}/ckpts/${project_name}/${exp_name}"}     # .../ckpts is the OBS folder where training outputs are saved, which can be customized.
TRAIN_FILE=${TRAIN_FILE:-"${OUTPUT_DIR}/dataset/gsm8k/precessed/train.parquet"}  # .../dataset/gsm8k is the OBS folder where the GSM8K training data is stored, which can be customized. /precessed/train.parquet is the preprocessed training data and does not need to be modified.
TEST_FILE=${TEST_FILE:-"${OUTPUT_DIR}/dataset/gsm8k/precessed/test.parquet"}  # .../dataset/gsm8k is the OBS folder where the GSM8K training data is stored, which can be customized. /precessed/test.parquet is the preprocessed test data and does not need to be modified.

python3 -m verl.trainer.main_ppo \
    --config-path=config \
    --config-name='ppo_megatron_trainer.yaml'\
    algorithm.adv_estimator=grpo \
    data.train_files="${TRAIN_FILE}" \
    data.val_files="${TEST_FILE}" \
    data.train_batch_size=32 \
    data.max_prompt_length=2048 \
    data.max_response_length=2048 \
    data.filter_overlong_prompts=True \
    data.truncation='error' \
    actor_rollout_ref.model.path=${MODEL_PATH} \
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
    actor_rollout_ref.actor.checkpoint.async_save=False \
    actor_rollout_ref.actor.optim.lr=1e-6 \
    actor_rollout_ref.model.use_remove_padding=True \
    actor_rollout_ref.actor.ppo_mini_batch_size=8 \
    actor_rollout_ref.actor.ppo_micro_batch_size_per_gpu=1 \
    actor_rollout_ref.actor.megatron.param_offload=${offload} \
    actor_rollout_ref.actor.megatron.optimizer_offload=${offload} \
    actor_rollout_ref.actor.megatron.grad_offload=${offload} \
    actor_rollout_ref.actor.megatron.pipeline_model_parallel_size=${train_pp} \
    actor_rollout_ref.actor.megatron.tensor_model_parallel_size=${train_tp} \
    actor_rollout_ref.actor.megatron.expert_model_parallel_size=8 \
    actor_rollout_ref.actor.megatron.expert_tensor_parallel_size=1 \
    actor_rollout_ref.actor.use_kl_loss=True \
    actor_rollout_ref.actor.kl_loss_coef=0.001 \
    actor_rollout_ref.actor.kl_loss_type=low_var_kl \
    actor_rollout_ref.actor.entropy_coeff=0 \
    actor_rollout_ref.actor.use_torch_compile=False \
    actor_rollout_ref.actor.policy_loss.loss_mode=gspo \
    actor_rollout_ref.rollout.log_prob_micro_batch_size_per_gpu=1 \
    actor_rollout_ref.rollout.tensor_model_parallel_size=${gen_tp} \
    actor_rollout_ref.rollout.name=vllm \
    actor_rollout_ref.rollout.gpu_memory_utilization=0.5 \
    actor_rollout_ref.rollout.n=4 \
    actor_rollout_ref.ref.use_torch_compile=False \
    actor_rollout_ref.ref.log_prob_micro_batch_size_per_gpu=1 \
    actor_rollout_ref.ref.megatron.pipeline_model_parallel_size=${train_pp} \
    actor_rollout_ref.ref.megatron.tensor_model_parallel_size=${train_tp} \
    actor_rollout_ref.ref.megatron.expert_model_parallel_size=4 \
    actor_rollout_ref.ref.megatron.expert_tensor_parallel_size=1 \
    actor_rollout_ref.ref.megatron.param_offload=${offload} \
    actor_rollout_ref.rollout.checkpoint_engine.update_weights_bucket_megabytes=4096 \
    reward.reward_manager.name=naive \
    actor_rollout_ref.ref.strategy=megatron \
    actor_rollout_ref.rollout.enforce_eager=True \
    ++actor_rollout_ref.ref.megatron.override_transformer_config.use_flash_attn=True \
    +actor_rollout_ref.actor.megatron.override_transformer_config.use_flash_attn=True \
    +actor_rollout_ref.actor.megatron.override_transformer_config.apply_rope_fusion=True \
    +actor_rollout_ref.actor.megatron.override_transformer_config.gradient_accumulation_fusion=True \

+actor_rollout_ref.actor.megatron.override_transformer_config.use_fused_ring_attention_update=True \
    +actor_rollout_ref.actor.megatron.override_transformer_config.use_distributed_optimizer=True \

+actor_rollout_ref.rollout.engine_kwargs.vllm.compilation_config.cudagraph_mode="FULL_AND_PIECE
WISE" \
    algorithm.use_kl_in_reward=False \
    trainer.critic_warmup=0 \
    trainer.logger='["console"]' \
    trainer.project_name=${project_name} \
    trainer.experiment_name=${exp_name} \
    trainer.n_gpus_per_node=${NGPUS_PER_NODE} \
    trainer.nnodes=${NNODES} \
    trainer.default_local_dir=${CKPTS_DIR} \
    trainer.resume_mode=auto \
    trainer.val_before_train=False \
    trainer.save_freq=50 \
    trainer.test_freq=50 \
    trainer.total_epochs=1 \
    trainer.total_training_steps=50 \
    actor_rollout_ref.rollout.calculate_log_probs=True \
    actor_rollout_ref.actor.router_replay.mode=R3 \
    actor_rollout_ref.rollout.enable_rollout_routing_replay=True 2>&1 | tee ${OUTPUT_DIR}/
run_qwen3_30b_a3b_npu_ma_r3.log

sleep 20s
if tail -n 100 ${OUTPUT_DIR}/run_qwen3_30b_a3b_npu_ma.log | grep -q 'Final validation metrics';
then
    exit 0
else
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
    exit 1
fi
```

3.2.5 Executing a Training Job

This section describes how to configure and execute a training job on the ModelArts training platform.

Prerequisites

- You have configured permissions on the ModelArts console. For details, see **Configuring Agency Authorization for ModelArts with One Click**. If you have not configured permissions, a message will be displayed on the top of the training job page, prompting you to configure permissions.
- The model weights, training data, data preprocessing script, and training script have been uploaded to an OBS bucket, for example, **obs://verl/verl-a2**.

Step 1: Configuring Basic Settings

1. Log in to the **ModelArts console** and choose **Model Build** > **Training**.
2. Click **Create Training Job**. Set the basic parameters of the training job according to the following table.

Table 3-17 Basic configuration

Parameter	Mandatory	Description	Example
Training Mode	Yes	The training mode can be **Fine-Tuning** or **Custom Job**. In this example, select **Custom Job**.	**Custom Job**
Name	Yes	(Mandatory) Name of the training job. The system automatically generates a name, which you can then rename.	Automatically generated
Description (Optional)	No	Job description, which helps you learn about the job information in the training job list.	**VeRL training**

Step 2: Defining Training Configuration

Set the key parameters of the training job according to **Table 3-18**.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Table 3-18 Training configuration

Parameter	Mandatory	Description	Example
Select Image	Yes	Select the container image required for training. In this example, select a preset image. The image environment required for VeRL framework training is preset on the platform and can be directly used.	**Preset Images**
Image address	Yes	Image path. The VeRL image contains the software required for training, such as the CANN package, driver, PyTorch, and VeRL framework code package.	**VeRL**

Boot Command	Yes	Command for starting the training image. The **cd** operation for switching to the local code directory in the container is mandatory.
		You need to upload the training job startup script and associated script files to your OBS bucket in advance.
		• Qwen3-8B requires the following files: **run_train_8b.sh**, **run_qwen3-8b_npu_ma.sh**, **gsm8k.py**
		• Qwen2.5-VL-32B-Instruct requires the following files: **run_train_32b.sh**, **run_qwen2_5_vl_32b_npu_ma.sh**, **geometry3k.py**
		• Qwen3-30B-A3B requires the following files: **run_train_30b.sh**, **run_qwen3-30b_a3b_npu_ma.sh**, **gsm8k.py**
			• Qwen3-8B
			`cd /home/ma-user/work/verl-a2`
			`bash /home/ma-user/work/verl-a2/run_train_8b.sh`
			• Qwen2.5-VL-32B-Instruct
			`cd /home/ma-user/work/verl-a2`
			`bash /home/ma-user/work/verl-a2/run_train_32b.sh`
			• Qwen3-30B-A3B
			`cd /home/ma-user/work/verl-a2`
			`bash /home/ma-user/work/verl-a2/run_train_30b.sh`

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Parameter	Mandatory	Description	Example
Code Directory	Yes	If the code is stored in OBS, select the code directory in the OBS bucket path.	**obs://verl/verl-a2**
Local Code Directory	Yes	Specify the local directory of the training container. When training starts, the system automatically downloads the code directory to this directory. Change the value to **/home/ma-user/work**, which is the same as the code directory set in the training script.	**/home/ma-user/work**
Environment Variable	No	The environment variables are already set in the training script, so you do not need to set them here.	N/A

Step 3: Configuring Resources

Set the resource configuration parameters of the training job according to **Table 3-19**.

Table 3-19 Resource configuration

Parameter	Mandatory	Description	Example
Source of resources	Yes	Public and dedicated resource pools are available for you to select. In this example, use a dedicated resource pool.	**Dedicated resource pool**
Resource Pool	Yes	Click **Select Resource Pool**. In the dialog box displayed on the right, select a physical resource pool with an 8 x Snt9b2 specification. The resource pool must be created in advance. Hover over **View** in the **Resource Fragment** column to check fragment details and check whether the resource pool meets the training requirements.	**Physical** 8 x Snt9b2

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Parameter	Mandatory	Description	Example
Specifications	Yes	Select an 8 x Snt9b2 specification.	8 x Snt9b2
Compute Nodes	Yes	Qwen3-8B model training requires 8-PU resources per node. Set this parameter to **1**. Qwen2.5-VL-32b-Instruct model training requires 16-PU resources. Set this parameter to **2**.	Qwen3-8B: **1** Qwen2.5-VL-32b-Instruct: **2** Qwen3-30b-A3B: **2**
Storage Mounting	Yes	Click **Add Extended Storage (OBS)** to store training artifacts.	**Extended Storage (OBS)**
		Directory: Select the training output path in the OBS bucket.	**obs://verl/verl-a2**
		Mount Path: Output path of the training artifacts in the training container.	**/output_dir**
		readonly: The mounted OBS path can only be read and cannot be written. The training artifacts need to be written to OBS. Do not select this option.	Do not select it.
Job Scheduling Priority	No	Scheduling priority of the training job.	Retain the default settings.
Preemption	No	When enabled, jobs that allow preemption may be terminated and re-queued if resource pool capacity is insufficient. To avoid losing training progress, configure resumable training before enabling this function.	Do not select it.

Step 4: Configuring HA

Set the HA configuration parameters of the training job according to **Table 3-20**.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Table 3-20 HA configuration

Parameter	Mandatory	Description	Example
Maximum Restarts	No	The system marks the job as failed if it continues to fail after reaching the maximum restart limit. The default value is **3**.	Retain the default settings.
Unconditional Auto Restart	No	As long as a training exception is detected, ModelArts unconditionally restarts the training job. To prevent invalid restarts, it supports a maximum of three consecutive unconditional restarts.	Select it.
Restart Upon Suspension	No	ModelArts continuously monitors job processes to detect suspension and optimize resource usage. When this feature is enabled, suspended jobs can be automatically restarted at the process level. To prevent unnecessary restarts, ModelArts limits consecutive restarts to three.	Select it.

Step 5: Managing Access Configuration

In this example, retain the default settings for parameters related to online debugging and remote SSH in the access configuration.

Step 6: Enabling Observability

In this example, retain the default settings for **Interconnect Metrics with AOM** in the observability configuration.

Step 7: Adjusting Additional Configurations

Set the additional configuration parameters of the training job according to **Table 3-21**.

Table 3-21 Additional configuration

Parameter	Mandatory	Description	Example
Persistent Log Saving	Yes	Logs will be deleted after 30 days. If you enable this function, logs will be saved to a specified OBS path. You can also download all logs on the job details page to a local path.	Select it. **obs://verl/verl-a2/logs/**

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Parameter	Mandatory	Description	Example
Job Visibility	No	By default, the job is visible to all users in the same workspace. **Creator**: The created training job is invisible to other users.	**Workspace**
Auto Stop	No	If this function is enabled, the job tracks its runtime. It stops automatically if the runtime surpasses the set time limit.	Do not select it.
Event Notification	No	If enabled, you will receive SMS or email notifications when specific events occur (such as job status changes or suspected suspension). These notifications may incur minor charges.	Do not select it.
Tags	No	To use the same tag for multiple cloud resources, set the tag in TMS.	Do not select it.

At the bottom of the **Create Training Job** page, click **Submit** to create the training job.

A training job runs for a period of time. You can go to the training job list or training details page to view the basic information about the training job.

After the training is complete, view the training logs and performance by referring to **Viewing the Training Output**. The training output is stored in the OBS bucket **obs://verl/verl-a2**.

For more information about ModelArts training, see **Model Training**.

3.2.6 Viewing the Training Output

The training output, such as preprocessed training data, checkpoints, logs, and model weights, is stored in the **obs://***<bucket_name>***/verl-a2** bucket.

The following shows the training result structure:

```
|——{OBS bucket}                                  # Custom bucket name, for example, verl
  |——{verl-a2}                                  # Custom OBS folder name, for example, verl-a2
    |——{ckpts}                                  # Custom OBS folder for storing checkpoints, for example,
ckpts
      |——{GRPO-Qwen3}                           # Custom OBS folder name, which is the project name,
for example, GRPO-Qwen3
        |——{GRPO-Qwen3-8B-npu}                  # Custom OBS folder, which is the experiment name,
for example, GRPO-Qwen3-8B-npu.
          |——latest_checkpointed_iteration.txt  # Iteration for saving the latest checkpoint
          |——global_step_{number}               # Training weight folder for step {number}
    |——run_qwen3-8b_npu_ma.log                  # Training log
    |——run_qwen2_5_vl_32b_npu_ma.log            # Training log
    |——run_qwen3_30b_a3b_npu_ma.log             # Training log
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Viewing Logs

Search for **obs://***<bucket_name>***/verl-a2/run_qwen3_8b_npu_ma.log**, **obs://***<bucket_name>***/verl-a2/run_qwen2_5_vl_32b_npu_ma.log**, or **obs://***<bucket_name>***/verl-a2/run_qwen3_30b_a3b_npu_ma.log** in your OBS bucket.

Checking Performance

Training performance is mainly checked based on two metrics in the training logs: throughput and convergence.

1. Throughput (tokens/s/p): The performance is calculated based on the **verl_grpo-***{af_model_name}***-***<Sequence length-Device type-Timestamp>***-run_log-N1-WS8.txt** file in the path specified by *${output_dir}*. The **perf/throughput** field in the training log file indicates the throughput of a single-step calculation. You can use the visualization tool **TrainingLogParser** to view the average training throughput.
2. Score convergence: The **critic/score/mean** value in the log approaches 1 and stabilizes over time during training iterations. You can also use the visualization tool **TrainingLogParser** to view the convergence of **critic/score/mean**.

3.3 Training the Qwen3 Series Models Adapted to the AReaL Framework on ModelArts

3.3.1 Overview

Description

ModelArts allows you to perform reinforcement learning on popular open-source models with the AReaL framework. Snt9b is used for training.

AReaL

AReaL is an open-source, fully asynchronous system for training large-scale inference and agent models. It is based on the ReaLHF project and follows open-source principles. We share all the details, data, and infrastructure needed to replicate our results and make the model freely available. AReaL helps anyone build their own AI agents easily and affordably.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Supported LLMs

Table 3-22 Supported LLMs

Model	Training Scenario	Training Framework	Megatron	PyTorch FSDP	Minimum PUs
Qwen3-8b	Reinforcement learning	AReaL	x	√	16 x Snt9b2
Qwen3-32b	Reinforcement learning	AReaL	x	√	24 x Snt9b2

Constraints

- Snt9b2 is used for training.

Resources and Files Required

Table 3-23 Resources and data files required for training

No.	Item	Mandatory	Description
1	ModelArts compute resources	Yes	You are advised to use Snt9b resources in a ModelArts dedicated resource pool. The pool must be created in advance.
2	OBS bucket	Yes	ModelArts requires you to create an OBS bucket for storing the model weight files, training data, training scripts, and training artifacts.
3	Model weight files	Yes	Model files for training. You must prepare them in advance. This document provides the download addresses.
4	Training datasets	Yes	Datasets for training. You must prepare them in advance. This document provides the download addresses.
5	Training scripts	Yes	Training job startup scripts. You must prepare them in advance. This document provides script examples, which can be directly used.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

No.	Item	Mandatory	Description
6	Training environment image	No	The training images are already set up on the platform. You do not need to prepare them. Just select an image when you create a training job. The image contains the software required for training, such as the CANN package, driver, PyTorch, and training framework code package.

Procedure

Table 3-24 Procedure description

Phase	Step	Description
Preparations	**Configuring Cloud Resources**	Prepare the dedicated resource pool and OBS bucket required for training.
	Preparing Model Weights and Training Data	Prepare the model weight files, training datasets, and data processing script files, and upload them to the OBS bucket.
	Preparing the Training Scripts	Prepare the script files required for training in advance and upload them to the OBS bucket.
Training	**Executing a Training Job**	Create a training job. During the creation, select a preset training environment image, select the resource specifications required for training, and set input and output parameters.
Viewing the training results	**Viewing the Training Output**	View the training logs and performance results.
	Viewing the Training Output	Obtain the trained model from the output path set in the training job.

3.3.2 Configuring Cloud Resources

The resources required for this guide are listed below. Prepare them in advance.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Resource Requirements

Table 3-25 Resources to be prepared

Item	Mandatory	Description	Reference
ModelArts compute resources	Yes	You are advised to use Snt9b resources in ModelArts dedicated resource pools. The dedicated resource pool must be created in advance. In addition, the dedicated resource pool must be able to access the external network through the VPC.	For details, see **Creating a Dedicated Resource Pool**.
OBS bucket	Yes	ModelArts requires you to create an OBS bucket for storing the model weight files, training data, training scripts, and training artifacts.	**Creating an OBS Bucket**

Creating an OBS Bucket

ModelArts requires you to create an OBS bucket for storing the model weight files, training data, training scripts, and training artifacts. For details, see **How Do I Create an OBS Bucket for ModelArts to Store Data?**

The following shows the folders in the OBS bucket. The planned directories must be the same as the file paths involved in the training script.

```
|——{OBS bucket}                # Custom bucket name, for example, your_bucket
  |——{areal}                 # Custom OBS folder name, for example, areal
    |——{models}              # Custom OBS folder for storing various models, for example, models
      |——{Qwen3-8B}          # Custom OBS folder for storing models, for example, Qwen3-8B
      |——{Qwen3-32B}         # Custom OBS folder for storing models, for example, Qwen3-32B
    |——a2_multi.sh.sh        # Training job startup script
    |——{dataset}/            # Custom OBS folder for storing various datasets, for example, dataset.
      |——{gsm8k}/            # OBS folder for storing the training dataset GSM8K
```

You can use tools such as **OBS Browser+** and **obsutil** to access and manage OBS buckets and upload and download files and data.

3.3.3 Preparing Model Weights and Training Data

Obtaining the Model Weights and Code

This section uses Qwen3 models as an example to describe the training process. **Table 3-26** lists the supported LLMs and the addresses for obtaining their weights.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Table 3-26 Supported LLMs and model weight sources

Series	Model	Training Scenario	Training Framework	Open-Source Weight File Link
Qwen3	Qwen3-8B	RLHF	AReaL	https://huggingface.co/Qwen/Qwen3-8B
	Qwen3-32B	RLHF	AReal	https://huggingface.co/Qwen/Qwen3-32B

NOTE

To download weight files from the Hugging Face website, set up a proxy. For instructions, search online.

Upload the downloaded model weight files to your OBS bucket. The following is an example of the directory for storing files in your OBS bucket based on the OBS plan:

```
obs://<bucket_name>/areal/models/Qwen3-8B
```

Obtaining the Dataset

GSM8K data is used for model training and does not need to be preprocessed.

1. Download the open-source dataset locally. The name of the directory where the dataset is stored must contain **gsm8k**. The following uses **gsm8k** as an example.

   Dataset link: https://huggingface.co/datasets/openai/gsm8k

2. Upload the dataset to **obs://***<bucket_name>***/areal/dataset**.

   ```
   obs://<bucket_name>/areal/dataset/gsm8k
   ```

3.3.4 Preparing the Training Scripts

This case study provides a sample training script, **a2_multi.sh**. You need to refer to this section to create your own .sh file and upload it to an OBS bucket.

Reinforcement learning configuration arguments appended to the python command in the script can be added directly to override the default values in **examples/math/gsm8k_grpo_npu.yaml**. For parameters that are not present in the YAML file but are actually valid, prepend a **+** to the parameter name to make it take effect. Descriptions for some of these configuration parameters are provided below. For more configuration parameters, see the **open-source AReaL YAML documentation**.

Explanation of Key YAML Configurations

Parameter	Description	Example Value
total_train_epochs	Total number of training epochs.	**10**

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Parameter	Description	Example Value
total_train_steps	Total number of training steps. The actual execution takes the smaller value between **total_train_epochs** ***steps_per_epoch** and **total_train_steps**. This parameter does not exist in the **gsm8k_grpo_npu.yaml** file by default; it can be specified in the script using **+total_train_steps=50**.	**50**
cluster.n_gpus_per_node	Number of NPUs per node.	**8**
cluster.n_nodes	Total number of nodes used.	**2**
gconfig.max_new_tokens	Maximum token length generated during inference.	**2048**
train_dataset.batch_size	Batch size for training data.	**256**
gconfig.n_samples	Number of samples to draw during inference sampling.	**4**
saver.freq_steps	Checkpoint saving frequency, measured in training steps.	**50**
save.freq_epochs	Checkpoint saving frequency, measured in training epochs.	**1**

Sample a2_multi.sh Script

Sample content:

```
#!/bin/bash
#set -e

ws="/home/ma-user/AReaL"
export HCCL_IF_BASE_PORT=62100
export HCCL_NPU_SOCKET_PORT_RANGE="62000-62050"
unset https_proxy http_proxy proxy ASCEND_RT_VISIBLE_DEVICES
source /usr/local/Ascend/nnal/atb/set_env.sh
# Change the values as needed.
NODE_NUM=${VC_WORKER_NUM:-1}
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
MASTER_ADDR="${VC_WORKER_HOSTS%%,*}"
DOMAIN_IP=$(python -c "import socket; print(socket.gethostbyname('${MASTER_ADDR}'))")
MASTER_NODE_IP="${DOMAIN_IP}"
MASTER_NODE_PORT="6699"
MODEL_PATH=${MODEL_PATH}
DATASET_PATH=${DATASET_PATH}
ALLOCATION_MODE=${ALLOCATION_MODE}
timestamp=$(date +%s)
GBS=${GBS:-128}
FILE_ROOT_PATH=${FILE_ROOT_PATH:-"/home/ma-user/areal_exp"}
CKPT_ROOT_PATH=${CKPT_ROOT_PATH:-"/home/ma-user/ckpts/areal_space"}
MAX_WAIT_SECONDS=900
CHECK_INTERVAL=3

get_current_ip() {
    local ip
    ip=$(ip -4 addr | grep -v '127.0.0.1' | grep -oP '(?<=inet\s)\d+(\.\d+){3}' | head -n1 2>/dev/null)
    if [[ -z "$ip" ]]; then
        ip=$(ip -4 addr | grep -v '127.0.0.1' | awk '/inet/ {gsub(/\/.*/,""); print $2}' | head -n1 2>/dev/null)
    fi
    echo "${ip:-}"
}

CURRENT_IP=$(get_current_ip)
if [[ -z "$CURRENT_IP" ]]; then
    echo "Failed to obtain the IP address of the current node."
    exit 1
fi
echo "current ip: $CURRENT_IP, master ip: $MASTER_NODE_IP"


if [ "$CURRENT_IP" = "$MASTER_NODE_IP" ]; then
    echo "start Ray master node..."
    ray stop --force
    ray start --head --port="$MASTER_NODE_PORT"

    elapsed_seconds=0
    echo "wait $NODE_NUM nodes..."
    while true; do
        current_node_num=$(ray status | grep -c "node_" || echo 0)
        if ! [[ "$current_node_num" =~ ^[0-9]+$ ]]; then
            current_node_num=0
        fi
        echo "current_node_num: $current_node_num"
        if [ "$current_node_num" -ge "$NODE_NUM" ]; then
            echo "all $NODE_NUM workers node is ready."
            break
        fi

        if [ "$elapsed_seconds" -ge "$MAX_WAIT_SECONDS" ]; then
            echo "timeout ($MAX_WAIT_SECONDS s), current_node_num: $current_node_num"
            ray stop --force
            exit 1
        fi
        sleep "$CHECK_INTERVAL"
        elapsed_seconds=$((elapsed_seconds + CHECK_INTERVAL))
    done


    echo "start job....."

    echo "args is MODEL_PATH=${MODEL_PATH} DATASET_PATH=${DATASET_PATH} GBS=${GBS}
FILE_ROOT_PATH=${FILE_ROOT_PATH} CKPT_ROOT_PATH=${CKPT_ROOT_PATH}"
    python -u -m areal.launcher.ray "${ws}/examples/math/gsm8k_rl.py" \
        --config "${ws}/examples/math/gsm8k_grpo_npu.yaml" rollout.dump_to_file=false \
        experiment_name=${timestamp}-gsm8k-grpo-multi cluster.fileroot=${FILE_ROOT_PATH} \
        trial_name=trial-0 saver.freq_steps=20 saver.fileroot=${CKPT_ROOT_PATH} recover.fileroot=$
{CKPT_ROOT_PATH} \
        allocation_mode=${ALLOCATION_MODE} \
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
        cluster.n_nodes="$NODE_NUM" \
        cluster.n_gpus_per_node=${MA_NUM_GPUS} cluster.name_resolve.type=ray  \
        gconfig.max_new_tokens=2048 \
        scheduler.type=ray \
        actor.path=${MODEL_PATH} \
        train_dataset.path=${DATASET_PATH} \
        valid_dataset.path=${DATASET_PATH} \
        train_dataset.batch_size=${GBS} | tee /home/ma-user/modelarts/log/latest.log

    sleep 60s
    tail -n 500 /home/ma-user/modelarts/log/latest.log | grep -q -E "Training completes|
JobState.COMPLETED" && exit 0 || exit 1


else
    echo "start workers, connect to master: $MASTER_NODE_IP:$MASTER_NODE_PORT"
    sleep 90s
    ray start \
        --address="${MASTER_NODE_IP}:${MASTER_NODE_PORT}"
fi
```

3.3.5 Executing a Training Job

This section describes how to configure and execute a training job on the ModelArts console.

Prerequisites

- You have configured permissions on the ModelArts console. For details, see **Configuring Agency Authorization for ModelArts with One Click**. If you have not configured permissions, a message will be displayed on the top of the training job page, prompting you to configure permissions.
- The model weights, training data, data preprocessing script, and training script have been uploaded to an OBS bucket, for example, **obs://areal/xxx**.

Step 1: Configuring Basic Settings

1. Log in to the **ModelArts console** and choose **Model Build** > **Training**.
2. Click **Create Training Job**. Set the basic parameters of the training job according to the following table.

Table 3-27 Basic configuration

Parameter	Mandatory	Description	Example
Training Mode	Yes	The training mode can be **Fine-Tuning** or **Custom Job**. In this example, select **Custom Job**.	**Custom Job**
Name	Yes	(Mandatory) Name of the training job. The system automatically generates a name, which you can then rename.	Automatically generated

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Parameter	Mandatory	Description	Example
Description (Optional)	No	Job description, which helps you learn about the job information in the training job list.	**AReaL training**

Step 2: Defining Training Configuration

Set the key parameters of the training job according to **Table 3-27**.

Note: When using OBS, it is recommended that you use either the **train.save_steps** or **train.save_epochs** parameter in the boot command and set the other parameter to **0**. VeOmni saves folders named **global_step_XX**. If **save_steps** and **save_epochs** have the same value, this can cause duplicate writes in OBS, leading to an I/O error.

For details about parameter settings, see **Arguments API Reference — VeOmni v0.1.0 documentation**.

Table 3-28 Training configuration

Parameter	Mandatory	Description	Example
Select Image	Yes	Select the container image required for training. In this example, select a preset image. The image environment required for VeOmni framework training is preset on the platform and can be directly used.	**Preset Images**
Image address	Yes	Image path. The VeOmni image contains the software required for training, such as the CANN package, driver, PyTorch, and VeOmni framework code package.	VeOmni

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Parameter	Mandatory	Description	Example

Boot Command	Yes	Boot command of the training image.
		You need to upload the training job startup script and associated script files to your OBS bucket in advance.
		After configuring all parameters in the script, run **train.sh**. This script contains configurations for multi-node communication, fine-tuning tasks, model saving, weight conversion, and checks.
		In the boot command, run the **cd** command to go to the Veomni directory.
			Multi-node training
			`cd /home/ma-user/VeOmni`
			`bash "${input_dir}/train.sh" /home/ma-user/VeOmni/tasks/train_torch.py /home/ma-user/VeOmni/configs/sft/qwen3_sft.yaml \`
			`--model.model_path "${model_dir}/Qwen3-32B" \`
			`--data.train_path "${dataset_dir}/tulu-first2000.parquet" \`
			`--train.data_parallel_mode fsdp2 \`
			`--train.init_device meta \`
			`--train.output_dir "${output_dir}/Qwen3-32B-Base-sft-tulu" \`
			`--train.save_epochs 3 \`
			`--train.num_train_epochs 3 \`
			`--train.ulysses_parallel_size 2 \`
			`--data.max_seq_len 4096 \`
			`--train.global_batch_size 8`
			The same script is used for single-node training.

Code Directory	Yes	If the code is in OBS, choose the OBS bucket path **obs://veomni/veomni-a2/**. During training, all files from the OBS path are copied to the training container's directory. Any changes made to this directory are not synced back to the OBS bucket. If the files in the OBS bucket are too big, downloading them to the training container can take a long time. This makes the job start slowly. It is recommended that the script, model folder, and dataset folder of a model training job be mounted to the same OBS bucket path and used only for this training job.	**obs://veomni/veomni-a2/**

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Parameter	Mandatory	Description	Example
Local Code Directory	Yes	Local directory of the training container. When the training starts, the system downloads the code directory from OBS to this directory and **overwrites** the existing content. Use a **newly created directory**. If the code directory in OBS is set to **obs://veomni/veomni-a2/** and the local code directory is set to **/home/ma-user/modelarts/user-job-dir**, the directory structure in the container is as follows: **/home/ma-user/modelarts/user-job-dir/veomni-a2/**	**/home/ma-user/modelarts/user-job-dir**

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Parameter	Mandatory	Description	Example
Environment Variable	No	When using a training script, add an output environment variable. For example, set **output_dir** as the parameter name (customizable). The output path **/home/ma-user/modelarts/user-job-dir/veomni/output/veomni-output** saves preprocessed data, checkpoints, and training logs to OBS. Set the environment variables for the model directory (**model_dir**), script directory (**input_dir**), and dataset directory (**dataset_dir**). Note: To use an extended OBS path, create a folder with the same name as your OBS **storage location** in the **cloud mount path** and then mount this folder. For example, if your OBS **storage location** is **obs://modelarts/Template/veomni/veomni-output/** and your **cloud mount path** is **/home/ma-user/modelarts/user-job-dir/veomni/output**, set the environment variable **output_dir** to **/home/ma-user/modelarts/user-job-dir/veomni/output/veomni-output**. This stores the output in OBS. To avoid confusion, use different names for the bottom-layer folders in OBS and the cloud mount path.	Parameter: **output_dir** Value: **/home/ma-user/modelarts/user-job-dir/veomni/output/veomni-output**

Step 3: Configuring Resources

Set the resource configuration parameters of the training job according to **Table 3-29**.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Table 3-29 Resource configuration

Parameter	Mandatory	Description	Example
Source of resources	Yes	Public and dedicated resource pools are available for you to select. In this example, use a dedicated resource pool.	**Dedicated resource pool**
Resource Pool	Yes	Click **Select Resource Pool**. In the dialog box displayed on the right, select a physical resource pool with an 8 x Snt9b2 specification. The resource pool must be created in advance. Hover over **View** in the **Resource Fragment** column to check fragment details and check whether the resource pool meets the training requirements.	**Physical** 8 x Snt9b2
Specifications	Yes	Select an 8 x Snt9b2 specification.	8 x Snt9b2
Compute Nodes	Yes	Qwen3-8B model training requires two nodes with 16-PU resources. Set this parameter to **2**. Qwen3-32B model training requires three nodes with 24-PU resources. Set this parameter to **3**.	Qwen3-8B: 2 Qwen3-32B: 3
Storage Mounting	No	Click **Add Extended Storage (OBS)** to store training artifacts.	**Extended Storage (OBS)**
		Directory: Select the training output path in the OBS bucket. The path must differ from the OBS code path.	**obs://<bucket_name>/ckpt_space**
		Mount Path: Output path of the training artifacts in the training container. Example: If **obs://***<bucket_name>*/*ckpt_space* is mounted to the container, the path is **/home/ma-user/ckpts/***ckpt_space*.	**/home/ma-user/ckpts**
		readonly: The mounted OBS path can only be read and cannot be written. The training artifacts need to be written to OBS. Do not select this option.	Do not select it.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Parameter	Mandatory	Description	Example
Job Scheduling Priority	No	Scheduling priority of the training job.	Retain the default settings.
Preemption	No	When enabled, jobs that allow preemption may be terminated and re-queued if resource pool capacity is insufficient. To avoid losing training progress, configure resumable training before enabling this function.	Do not select it.

Step 4: Configuring HA

Set the HA configuration parameters of the training job according to **Table 3-30**.

Table 3-30 HA configuration

Parameter	Mandatory	Description	Example
Maximum Restarts	No	The system marks the job as failed if it continues to fail after reaching the maximum restart limit. The default value is **3**.	Retain the default settings.
Unconditional Auto Restart	No	As long as a training exception is detected, ModelArts unconditionally restarts the training job. To prevent invalid restarts, it supports a maximum of three consecutive unconditional restarts.	Do not select it.
Restart Upon Suspension	No	ModelArts continuously monitors job processes to detect suspension and optimize resource usage. When this feature is enabled, suspended jobs can be automatically restarted at the process level. To prevent unnecessary restarts, ModelArts limits consecutive restarts to three.	Select it.

Step 5: Managing Access Configuration

In this example, retain the default settings for parameters related to online debugging and remote SSH in the access configuration.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Step 6: Enabling Observability

In this example, retain the default settings for **Interconnect Metrics with AOM** in the observability configuration.

Step 7: Adjusting Additional Configurations

Set the additional configuration parameters of the training job according to **Table 3-31**.

Table 3-31 Additional configuration

Parameter	Mandatory	Description	Example
Persistent Log Saving	Yes	Logs will be deleted after 30 days. If you enable this function, logs will be saved to a specified OBS path. You can also download all logs on the job details page to a local path. The path can be configured as needed.	Select it.
Job Visibility	No	By default, the job is visible to all users in the same workspace. **Creator**: The created training job is invisible to other users.	**Workspace**
Auto Stop	No	If this function is enabled, the job tracks its runtime. It stops automatically if the runtime surpasses the set time limit.	Do not select it.
Event Notification	No	If enabled, you will receive SMS or email notifications when specific events occur (such as job status changes or suspected suspension). These notifications may incur minor charges.	Do not select it.
Tags	No	To use the same tag for multiple cloud resources, set the tag in TMS.	Do not select it.

At the bottom of the **Create Training Job** page, click **Submit** to create the training job.

A training job runs for a period of time. You can go to the training job list or training details page to view the basic information about the training job.

After the training is complete, view the training logs and performance by referring to **Viewing the Training Output**.

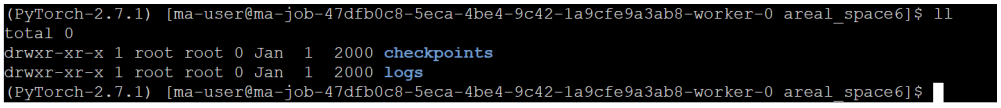
 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

For more information about ModelArts training, see **Model Training**.

3.3.6 Viewing the Training Output

The training output structure is below (Note: If **CKPT_ROOT_PATH** is configured, checkpoints will be stored in that path. AReaL logs are located under **FILE_ROOT_PATH**, which defaults to **/home/ma-user/areal_exp**).

ModelArts platform logs can be viewed directly on the platform UI page.



Viewing Logs

Check the epoch information and **ppo_actor/task_reward/avg** in the platform logs. Download the logs to create the reward curve.

Checking Performance

Filter the logs with **timeperf** to see how long each module takes to run.

3.3.7 Debugging a Training Job

When executing a production training job in ModelArts, any execution error will cause the job to exit immediately. This makes it difficult to inspect the container state for troubleshooting and prevents quick retries when configurations (such as training startup parameters or environment variables) are incorrect, causing significant inconvenience.

Therefore, it is recommended that you use the following method during the initial adaptation phase to retain the container for rapid troubleshooting and retries.

```
# The last line of the boot command
sleep 1h
```

For details about how to use CloudShell to debug a training job, see **Using Cloud Shell to Debug a Production Training Job**.

FAQ:

How do I use Cloud Shell to view the environment variables injected into the user process by the ModelArts training platform?

Running the **env** command in Cloud Shell only returns the container's default environment variables, which may differ from those of the actual user process. To inspect the correct variables, use the following method instead:

1. Check the user process ID (PID):
   ```
   ps -aef
   ```
2. View the environment variables of that specific user process:
   ```
   cat /proc/$PID/environ | tr '\0' '\n'
   ```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

4 Video Generation Model Training and Inference

4.1 Inference Guide for Wan2.1, Wan2.2, HunyuanVideo, and CogVideo Series Models Adapted to NPU via ModelArts Lite Server (6.5.911)

Solution Overview

Many companies now explore advanced AI video generation models to boost content creation speed. Yet, limited compute often makes running these models locally challenging. To solve this, businesses can use ModelArts Lite Server with NPUs for efficient model deployment, speeding up content production.

This section describes how to use NPUs to perform text-to-video inference, image-to-video inference, and text-to-image inference using Wan2.1, Wan2.2, HunyuanVideo, and CogVideo video generation models in ModelArts Lite Server. **Table 4-1** lists the supported models.

Table 4-1 Supported models

Category	Model	Scenario
Wan2.1	Wan2.1-T2V-14B-Diffusers	Text-to-video/Image
	Wan2.1-T2V-1.3B-Diffusers	Text-to-video
	Wan2.1-I2V-14B-480P-Diffusers	Image-to-video
	Wan2.1-I2V-14B-720P-Diffusers	Image-to-video
Wan2.2	Wan2.2-T2V-A14B-Diffusers	Text-to-video

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Category	Model	Scenario
	Wan2.2-I2V-A14B-Diffusers	Image-to-video
HunyuanVideo	HunyuanVideo	Text-to-video
CogVideo	CogVideoX-5b	Text-to-video

Resource Specifications

Use Snt9B or Snt9B23 single-node resources in the Lite Server environment.

Table 4-2 Snt9B23 environment requirements

Name	Version
Driver	25.2.1
PyTorch	pytorch_2.5.1

Table 4-3 Snt9B environment requirements

Name	Version
Driver	24.1.0.6
PyTorch	pytorch_2.5.1

Obtaining the Software Package and Images

Table 4-4 Software package and images to be obtained

Category	Name	How to Obtain
Plug-in code package	AscendCloud-AIGC-6.5.911-xxx.zip. **NOTE** *xxx* in the package name indicates the timestamp, which is subject to the actual package release time.	Download ModelArts 6.5.911.3 from **Support-E**. **NOTE** If the software information does not appear when opening the download link, you lack access permissions. Contact your company's Huawei technical support for assistance with downloading.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Category	Name	How to Obtain
Base image	Snt9B: CN Southwest-Guiyang1 swr.cn-southwest-2.myhuaweicloud.com/atelier/pytorch_ascend:pytorch_2.5.1-cann_8.2.rc2-py_3.11-hce_2.0.2509-aarch64-snt9b-20251105194139-a70c13c Snt9B23: CN Southwest-Guiyang1 swr.cn-southwest-2.myhuaweicloud.com/atelier/pytorch_ascend:pytorch_2.5.1-cann_8.2.rc2-py_3.11-hce_2.0.2509-aarch64-snt9b23-20251106111344-87d9583	Pull the image from SWR.

Constraints

- This document applies to ModelArts 6.5.911. Obtain the required software package and image by referring to **Table 4-4**. Follow the version mapping when using this document.
- Ensure that the container can access the Internet.

Step 1: Preparing the Environment

1. **Enable Lite Server resources** and obtain passwords. Verify SSH access to all servers. Confirm proper network connectivity between them.

   NOTE

   If a container is used or shared by multiple users, you should restrict the container from accessing the OpenStack management address (169.254.169.254) to prevent host machine metadata acquisition. For details, see **Forbidding Containers to Obtain Host Machine Metadata**.

2. Log in to the server via SSH and check the NPUs. Obtain the NPU device information:

   ```
   npu-smi info                    # Run this command on each instance node to view the NPU status.
   npu-smi info -l | grep Total    # Run this command on each instance node to view the total number of PUs.
   ```

   If an error occurs, the NPU devices on the server may not be properly installed, or the NPU image may be mounted to another container. **Install the firmware and driver** or release the mounted NPUs.

3. Check whether Docker is installed.

   ```
   docker -v   # Check whether Docker is installed.
   ```

   If Docker is not installed, run this command:

   ```
   yum install -y docker-engine.aarch64 docker-engine-selinux.noarch docker-runc.aarch64
   ```

4. Configure IP forwarding for intra-container network accesses. Run the command below to check the value of **net.ipv4.ip_forward**. Skip this step if the value is **1**.

   ```
   sysctl -p | grep net.ipv4.ip_forward
   ```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

If the value is not **1**, configure IP forwarding:

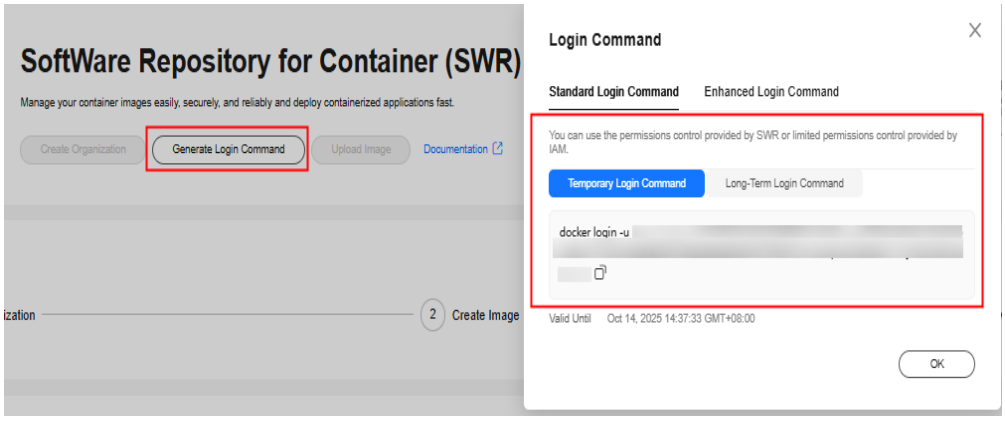
```
sed -i 's/net\.ipv4\.ip_forward=0/net\.ipv4\.ip_forward=1/g' /etc/sysctl.conf
sysctl -p | grep net.ipv4.ip_forward
```

Step 2: Obtaining the Base Image

Use official images to deploy inference services. For details about the image path *{image_url}*, see **Table 4-4**.

```
docker pull {image_url}
```

To log in to the SWR console, log in to the **SWR console** and obtain the login command by referring to the figure below.



Step 3: Starting the Container Image

1. Start the container image. Before starting the container, modify the parameters in *${}* according to the parameter description.

   Start the Snt9B23 container:

   ```
   export work_dir="Custom mounted working directory"
   export container_work_dir="Custom working directory mounted to the container"
   export container_name="Custom container name"
   export image_name="Image name or ID"
   // Start a container to run the image.
   docker run  -itd --net=host \
      --privileged \
      --device=/dev/davinci_manager \
      --device=/dev/devmm_svm \
      --device=/dev/hisi_hdc \
      --shm-size=256g \
      -v /usr/local/dcmi:/usr/local/dcmi \
      -v /usr/local/Ascend/driver:/usr/local/Ascend/driver \
      -v /var/log/npu/:/usr/slog \
      -v /usr/local/sbin/npu-smi:/usr/local/sbin/npu-smi \
      -v ${work_dir}:${container_work_dir} \
      --name ${container_name} \
      ${image_name} \
      /bin/bash
   ```

   Start the Snt9B container:

   ```
   export work_dir="Custom mounted working directory"
   export container_work_dir="Custom working directory mounted to the container"
   export container_name="Custom container name"
   export image_name="Image name or ID"
   // Start a container to run the image.
   docker run  -itd --net=bridge \
      --device=/dev/davinci0 \
   ```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
--device=/dev/davinci1 \
--device=/dev/davinci2 \
--device=/dev/davinci3 \
--device=/dev/davinci4 \
--device=/dev/davinci5 \
--device=/dev/davinci6 \
--device=/dev/davinci7 \
--device=/dev/davinci_manager \
--device=/dev/devmm_svm \
--device=/dev/hisi_hdc \
--shm-size=256g \
-v /usr/local/dcmi:/usr/local/dcmi \
-v /usr/local/Ascend/driver:/usr/local/Ascend/driver \
-v /var/log/npu/:/usr/slog \
-v /usr/local/sbin/npu-smi:/usr/local/sbin/npu-smi \
-v ${work_dir}:${container_work_dir} \
--name ${container_name} \
${image_name} \
/bin/bash
```

Parameters:

- **-v** *${work_dir}*:*${container_work_dir}*: host directory to be mounted to the container. The host and container use different file systems. *work_dir* indicates the working directory on the host. The directory stores files such as code and data required for the project. *container_work_dir* indicates the directory to be mounted to the container. The two paths can be the same.

  NOTE

  - The **/home/ma-user** directory cannot be mounted to the container. This directory is the home directory of user **ma-user**. If the container is mounted to **/home/ma-user**, the container conflicts with the base image when being started. As a result, the base image is unavailable.
  - Both the driver and npu-smi must be mounted to the container.
- **--name** *${container_name}*: container name, which is used when you access the container. You can define a container name.
- *${image_name}*: name of the base image of the corresponding model. For details, see **Table 4-4**.
- **--device=/dev/davinci0**: mount the corresponding PU to the container. If multiple PUs need to be mounted, add the configuration items one by one.

2. Access the container through the container name.

   Log in to the Snt9B23 as user **root**.

   ```
   docker exec -it -u root ${container_name} bash
   ```

   For Snt9B, the **ma-user** user is used by default. All the subsequent operations are performed as user **ma-user**.

   ```
   docker exec -it ${container_name} bash
   ```

Step 4: Installing Dependencies and Software Packages

1. To use **git clone** and **git lfs** commands to download large models, see the following operations:

   a. Enter the URL below in the browser to download the **git-lfs** package and upload it to the **/home/ma-user** directory of the container.

   ```
   https://github.com/git-lfs/git-lfs/releases/download/v3.2.0/git-lfs-linux-arm64-v3.2.0.tar.gz
   ```

   Alternatively, download **git-lfs** to the container for direct use.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
cd /home/ma-user
wget https://github.com/git-lfs/git-lfs/releases/download/v3.2.0/git-lfs-linux-arm64-v3.2.0.tar.gz
```

b. Go to the container and run the **git-lfs** installation commands.

```
cd /home/ma-user
tar -zxvf git-lfs-linux-arm64-v3.2.0.tar.gz
cd git-lfs-3.2.0
sudo sh install.sh
```

c. Disable SSL verification for Git configuration.

```
git config --global http.sslVerify false
```

2. Install the AscendX_Video software package.

a. Upload the AscendX_Video software package **AscendCloud-AIGC-*.zip** to the **/home/ma-user** directory of the container. For details about how to obtain the package, see **Obtaining the Software Package and Images**.

b. Decompress the **AscendCloud-AIGC-*.zip** file, and run the following commands to install the Python dependencies:

```
cd /home/ma-user
unzip AscendCloud-AIGC-*.zip -d ./AscendCloud
cp -r /home/ma-user/AscendCloud/aigc_inference/torch_npu/ascendx_video ./
cd /home/ma-user/ascendx_video
pip install seal-*-linux_aarch64.whl
pip install check_device-*-linux_aarch64.whl
pip install ascendx_video-*-none-any.whl
```

c. Install the operator environment.

If the Snt9B23 machine is used, run the following command:

```
cd /home/ma-user/AscendCloud/opp/A3
```

If the Snt9B machine is used, run the following command:

```
cd /home/ma-user/AscendCloud/opp/A2
```

Install the operator:

```
unzip AscendCloud-OPP-*.zip
unzip AscendCloud-OPP-*-torch-2.5.1-py311-*.zip -d ./AscendCloud_OPP
cd AscendCloud_OPP
pip install *.whl
mkdir -p /home/ma-user/operate
bash ./ascend_cloud_ops_ascend_turbo-*_linux_aarch64.run --install-path=/home/ma-user/operate
bash ./ascend_cloud_ops_custom_opp-*_linux_aarch64_ascend910b_ascend910_93.run --install-path=/home/ma-user/operate
cd ..
unzip AscendCloud-OPS-ADV-*.zip -d ./AscendCloud_OPS-ADV
cd AscendCloud_OPS-ADV
bash ./CANN-custom_ops-*-linux.aarch64.run --install-path=/home/ma-user/operate
```

3. Initialize environment variables.

Note that the environment needs to be initialized each time you access the container.

```
source /home/ma-user/operate/AscendTurbo/set_env.bash
source /home/ma-user/operate/vendors/customize/bin/set_env.bash
source /home/ma-user/operate/vendors/customize_cloud/bin/set_env.bash
```

Step 5: Downloading Model Weights

Download the weight file to the container directory. The following lists the model addresses.

- Wan-AI/Wan2.1-T2V-14B-Diffusers: **https://huggingface.co/Wan-AI/Wan2.1-T2V-14B-Diffusers**

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

- Wan-AI/Wan2.1-T2V-1.3B-Diffusers: **https://huggingface.co/Wan-AI/Wan2.1-T2V-1.3B-Diffusers**
- Wan-AI/Wan2.1-I2V-14B-480P-Diffusers: **https://huggingface.co/Wan-AI/Wan2.1-I2V-14B-480P-Diffusers**
- Wan-AI/Wan2.1-I2V-14B-720P-Diffusers: **https://huggingface.co/Wan-AI/Wan2.1-I2V-14B-720P-Diffusers**
- Wan-AI/Wan2.2-T2V-A14B-Diffusers: **https://huggingface.co/Wan-AI/Wan2.2-T2V-A14B-Diffusers**
- Wan-AI/Wan2.2-I2V-A14B-Diffusers: **https://huggingface.co/Wan-AI/Wan2.2-I2V-A14B-Diffusers**
- HunyuanVideo: **https://huggingface.co/tencent/HunyuanVideo**
- Cogvideo: **https://huggingface.co/THUDM/CogVideoX-5b**

Save the weights to the **/home/ma-user/ascendx_video/weights** directory, for example:

```
weights
└──Wan-AI
   ├──Wan2.1-I2V-14B-480P-Diffusers
   ├──Wan2.1-I2V-14B-720P-Diffusers
   ├──Wan2.1-T2V-14B-Diffusers
   ├──Wan2.1-T2V-1.3B-Diffusers
   ├──Wan2.2-I2V-A14B-Diffusers
   └──Wan2.2-T2V-A14B-Diffusers
├──HunyuanVideo
└──CogVideoX-5b
```

Step 6: Performing Inference Using the Wan2.1 Text-to-Video Model

The following scripts are stored in the **/home/ma-user/ascendx_video/scripts/** directory:

- **infer_wan2.1_14b_t2v_480p.sh**: 480P inference script of the Wan text-to-video model Wan2.1-T2V-14B.
- **infer_wan2.1_14b_t2v_720p.sh**: 720P inference script of the Wan text-to-video model Wan2.1-T2V-14B.
- **infer_wan2.1_1.3b_t2v.sh**: inference script of the Wan text-to-video model Wan2.1-T2V-1.3B.

Run the commands below to start the inference task. The following uses **infer_wan2.1_14b_t2v_480p.sh** as an example.

```
cd /home/ma-user/ascendx_video/scripts/
bash infer_wan2.1_14b_t2v_480p.sh
```

The following describes the parameters of the text-to-video inference script **infer_wan2.1_14b_t2v_480p.sh**. The parameters of scripts **infer_wan2.1_1.3b_t2v.sh** and **infer_wan2.1_14b_t2v_720p.sh** are similar to those of **infer_wan2.1_14b_t2v_480p.sh**.

```
export MASTER_ADDR=127.0.0.1
export MASTER_PORT=29505

export PYTORCH_NPU_ALLOC_CONF=expandable_segments:True
export MEMORY_FRAGMENTATION=1
export COMBINED_ENABLE=1
export TASK_QUEUE_ENABLE=2
export TOKENIZERS_PARALLELISM=false
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
export ASCEND_RT_VISIBLE_DEVICES=0,1,2,3,4,5,6,7
N_NPUS=8
torchrun --nproc_per_node=$N_NPUS --master_addr $MASTER_ADDR --master_port $MASTER_PORT ../
infer.py \
    --model Wan2.1-T2V-14B \
    --pretrained_model_name_or_path "../weights/Wan-AI/Wan2.1-T2V-14B-Diffusers" \
    --save_path ./output.mp4 \
    --num_inference_steps 50 \
    --width 832 \
    --height 480 \
    --frames 81 \
    --sp $N_NPUS \
    --fsdp \
    --vae_lightning \
    --turbo_mode faiz \
    --atten_a8w8 \
    --matmul_a8w8 \
    --rope_fused \
    --seed 42 \
    --prompt "A young boy with short brown hair, dressed in a dark blue t-shirt and red pants, is seen
playing a KAWAI upright piano with skill and concentration. The piano's glossy black surface reflects the
room's lighting, and its white and black keys are arranged in a standard layout, indicating a scene of
musical practice or learning. The boy's hands move over the keys, suggesting he is engaged in playing or
practicing a piece." \
    --negative_prompt "vivid tone, overexposure, static, blurred details, subtitle, style, work, painting,
picture, static, overall gray, worst quality, low quality, JPEG compression residue, ugly, incomplete,
redundant fingers, poorly drawn hands, poorly drawn face, deformed, disfigured, deformed limbs, finger
fusion, static picture, messy background, three legs, many people in the background, walking backward"
```

- **ASCEND_RT_VISIBLE_DEVICES**: ID of the used PU.
- **N_NPUS**: number of used PUs. You are advised to use eight PUs.
- **model**: supported inference model. Currently, Wan2.1-T2V-14B, Wan2.1-I2V-14B, Wan2.1-T2V-1.3B, Wan2.2-T2V-A14B, and Wan2.2-I2V-A14B are supported.
- **pretrained_model_name_or_path**: weight address of the corresponding model.
- **save_path**: path for storing the video generated during inference.
- **num_inference_steps**: number of inference steps.
- **frames, height, width**: dimensions of the generated video, including the number of frames, height, and width. Currently, 81 x 480 x 832, 121 x 480 x 832, 81 x 720 x 1280, and 121 x 720 x 1280 are supported.
- **prompt, negative_prompt**: positive and negative prompts for generating a video.
- **sp**: sequence parallelism parameter. It is recommended that the value be the same as the number of inference PUs.
- **fsdp**: data parallelism. **None**, **all**, **text_encoder**, and **transformer** are supported. The default value is **None**, indicating that the function is disabled. If this function is enabled, the default value is **all**, indicating that parallelism is enabled for **text_encoder** and **transformer**. If this parameter is set to **text_encoder** or **transformer**, parallelism is enabled only for the specified module.
- **vae_lightning**: VAE acceleration. This parameter is supported only in the multi-PU scenario. If this parameter is not set, VAE acceleration is disabled. Enabling this function can improve VAE performance.
- **turbo_mode**: acceleration mode. **default** and **faiz** are supported. The default value is **default**, indicating that the function is disabled. **faiz** is recommended

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

for high performance. If this parameter is not set, the acceleration mode is disabled. Enabling this function can accelerate video inference, but slightly affects the accuracy.

- **atten_a8w8**: atten quantization acceleration. Set this parameter for high performance. If this parameter is not set, atten quantization acceleration is disabled. Enabling this function can accelerate video inference, but slightly affects the accuracy.
- **matmul_a8w8**: matmul quantization acceleration. Set this parameter for high performance. If this parameter is not set, matmul quantization acceleration is disabled. Enabling this function can accelerate video inference, but slightly affects the accuracy.
- **rope_fused**: rotary position encoding fusion operator. Set this parameter for high performance. If this parameter is not set, the fusion operator is disabled. Enabling this function can accelerate video inference, but slightly affects the accuracy.
- **seed**: Random number seed. The default value is **42**, which affects the effect of the generated image.

After the inference task is complete, the generated video file **output.mp4** is stored in the **save_path** directory, and the script is stored in the **/home/ma-user/ascendx_video/scripts** directory by default. From there, view the inference result.

Step 7: Performing Inference Using the Wan2.1 Image-to-Video Model

Before starting inference using the image-to-video model, download the sample image and save it to the **/home/ma-user/ascendx_video/scripts** directory.

Figure 4-1 Example



The following scripts are stored in the **/home/ma-user/ascendx_video/scripts/** directory:

- **infer_wan2.1_14b_i2v_480p.sh**: 480P inference script of the Wan2.1-I2V-14B image-to-video model.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

- **infer_wan2.1_14b_i2v_720p.sh**: 720P inference script of the Wan2.1-I2V-14B image-to-video model.

Run the commands below to start the inference task. The following uses **infer_wan2.1_14b_i2v_480p.sh** as an example.

```
cd /home/ma-user/ascendx_video/scripts/
bash infer_wan2.1_14b_i2v_480p.sh
```

The following describes the parameters of the image-to-video inference script **infer_wan2.1_14b_i2v_480p.sh**. The parameters of script **infer_wan2.1_14b_i2v_720p.sh** are similar to those of **infer_wan2.1_14b_i2v_480p.sh**.

```
export MASTER_ADDR=127.0.0.1
export MASTER_PORT=29505

export PYTORCH_NPU_ALLOC_CONF=expandable_segments:True
export MEMORY_FRAGMENTATION=1
export COMBINED_ENABLE=1
export TASK_QUEUE_ENABLE=2
export TOKENIZERS_PARALLELISM=false
export ASCEND_RT_VISIBLE_DEVICES=0,1,2,3,4,5,6,7
N_NPUS=8
torchrun --nproc_per_node=$N_NPUS --master_addr $MASTER_ADDR --master_port $MASTER_PORT ../
infer.py \
    --model Wan2.1-I2V-14B \
    --pretrained_model_name_or_path "../weights/Wan-AI/Wan2.1-I2V-14B-480P-Diffusers" \
    --task_type i2v \
    --i2v_image_path ./astronaut.jpg \
    --save_path ./output.mp4 \
    --num_inference_steps 40 \
    --width 832 \
    --height 480 \
    --frames 81 \
    --sp $N_NPUS \
    --fsdp \
    --vae_lightning \
    --turbo_mode faiz \
    --atten_a8w8 \
    --matmul_a8w8 \
    --rope_fused \
    --seed 42 \
    --prompt "An astronaut hatching from an egg, on the surface of the moon, the darkness and depth
of space realised in the background. High quality, ultrarealistic detail and breath-taking movie-like camera
shot." \
    --negative_prompt "vivid tone, overexposure, static, blurred details, subtitle, style, work, painting,
picture, static, overall gray, worst quality, low quality, JPEG compression residue, ugly, incomplete,
redundant fingers, poorly drawn hands, poorly drawn face, deformed, disfigured, deformed limbs, finger
fusion, static picture, messy background, three legs, many people in the background, walking backward"
```

- **task_type**: inference task type. The value can be **t2v** (text-to-video), **i2v** (image-to-video), or **t2i** (text-to-image). The default value is **i2v**.
- **i2v_image_path**: path of the image used for video generation.
- For other parameters, use the same settings as those of **infer_wan_14b_t2v_480p.sh**. For details, see **Step 6: Performing Inference Using the Wan2.1 Text-to-Video Model**.

After the inference task is complete, the generated video file **output.mp4** is stored in the **save_path** directory, and the script is stored in the **/home/ma-user/ascendx_video/scripts** directory by default. From there, view the inference result.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Step 8: Performing Inference Using the Wan2.1 Text-to-Image Model

The following scripts are stored in the **/home/ma-user/ascendx_video/scripts/** directory:

- **infer_wan2.1_14b_t2i_480p.sh**: 480P inference script of the Wan2.1-T2V-14B text-to-image model.
- **infer_wan2.1_14b_t2i_720p.sh**: 720P inference script of the Wan2.1-T2V-14B text-to-image model.

Run the commands below to start the inference task. The following uses **infer_wan2.1_14b_t2i_480p.sh** as an example.

```
cd /home/ma-user/ascendx_video/scripts/
bash infer_wan2.1_14b_t2i_480p.sh
```

The following describes the parameters of the text-to-image inference script **infer_wan2.1_14b_t2i_480p.sh**. The parameters of script **infer_wan2.1_14b_t2i_720p.sh** are similar to those of **infer_wan2.1_14b_t2i_480p.sh**.

```
export MASTER_ADDR=127.0.0.1
export MASTER_PORT=29505

export PYTORCH_NPU_ALLOC_CONF=expandable_segments:True
export MEMORY_FRAGMENTATION=1
export COMBINED_ENABLE=1
export TASK_QUEUE_ENABLE=2
export TOKENIZERS_PARALLELISM=false
export ASCEND_RT_VISIBLE_DEVICES=0
N_NPUS=1
torchrun --nproc_per_node=$N_NPUS --master_addr $MASTER_ADDR --master_port $MASTER_PORT ../
infer.py \
        --model Wan2.1-T2V-14B \
        --pretrained_model_name_or_path "../weights/Wan-AI/Wan2.1-T2V-14B-Diffusers" \
        --task_type t2i \
        --save_path ./output.png \
        --num_inference_steps 40 \
        --width 832 \
        --height 480 \
        --frames 1 \
        --atten_a8w8 \
        --matmul_a8w8 \
        --rope_fused \
        --seed 42 \
        --prompt "An astronaut hatching from an egg, on the surface of the moon, the darkness and depth
of space realised in the background. High quality, ultrarealistic detail and breath-taking movie-like camera
shot." \
        --negative_prompt "vivid tone, overexposure, static, blurred details, subtitle, style, work, painting,
picture, static, overall gray, worst quality, low quality, JPEG compression residue, ugly, incomplete,
redundant fingers, poorly drawn hands, poorly drawn face, deformed, disfigured, deformed limbs, finger
fusion, static picture, messy background, three legs, many people in the background, walking backward"
```

The parameters are the same as those of **infer_wan_14b_t2v.sh**. For details, see **Step 6: Performing Inference Using the Wan2.1 Text-to-Video Model**.

After the inference task is complete, the generated image file **output.png** is stored in the **save_path** directory, and the script is stored in the **/home/ma-user/ascendx_video/scripts** directory by default. From there, view the inference result.

Step 9: Performing Inference Using the Wan2.2 Text-to-Video Model

The following scripts are stored in the **/home/ma-user/ascendx_video/scripts/** directory:

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

- **infer_wan2.2_14b_t2v_480p.sh**: 480P inference script of the Wan text-to-video model Wan2.2-T2V-A14B-Diffusers.
- **infer_wan2.2_14b_t2v_720p.sh**: 720P inference script of the Wan text-to-video model Wan2.2-T2V-A14B-Diffusers.

Run the commands below to start the inference task. The following uses **infer_wan2.2_14b_t2v_480p.sh** as an example.

```
cd /home/ma-user/ascendx_video/scripts/
bash infer_wan2.2_14b_t2v_480p.sh
```

The following describes the parameters of the text-to-video inference script **infer_wan2.2_14b_t2v_480p.sh**. The parameters of script **infer_wan2.2_14b_t2v_720p.sh** are similar to those of **infer_wan2.2_14b_t2v_480p.sh**.

```
export MASTER_ADDR=127.0.0.1
export MASTER_PORT=29505

export PYTORCH_NPU_ALLOC_CONF=expandable_segments:True
export MEMORY_FRAGMENTATION=1
export COMBINED_ENABLE=1
export TASK_QUEUE_ENABLE=2
export TOKENIZERS_PARALLELISM=false
export ASCEND_RT_VISIBLE_DEVICES=0,1,2,3,4,5,6,7
N_NPUS=8
torchrun --nproc_per_node=$N_NPUS --master_addr $MASTER_ADDR --master_port $MASTER_PORT ../infer.py \
        --model Wan2.2-T2V-A14B \
        --pretrained_model_name_or_path ../weights/Wan-AI/Wan2.2-T2V-A14B-Diffusers \
        --task_type t2v \
        --save_path ./output.mp4 \
        --num_inference_steps 40 \
        --width 832 \
        --height 480 \
        --frames 81 \
        --sp $N_NPUS \
        --fsdp text_encoder \
        --vae_lightning \
        --inf_vram_blocks_num 1 \
        --vae_lightning \
        --atten_a8w8 \
        --matmul_a8w8 \
        --rope_fused \
        --guidance_scale 3.0 \
        --guidance_scale_2 4.0 \
        --seed 42 \
        --prompt "An astronaut hatching from an egg, on the surface of the moon, the darkness and depth of space realised in the background. High quality, ultrarealistic detail and breath-taking movie-like camera shot." \
        --negative_prompt "vivid tone, overexposure, static, blurred details, subtitle, style, work, painting, picture, static, overall gray, worst quality, low quality, JPEG compression residue, ugly, incomplete, redundant fingers, poorly drawn hands, poorly drawn face, deformed, disfigured, deformed limbs, finger fusion, static picture, messy background, three legs, many people in the background, walking backward"
```

- **inf_vram_blocks_num**: GPU memory optimization. Currently, only **1** is supported. If this function is enabled, the **fsdp text_encoder** parameter is required.
- **guidance_scale**: no classifier guidance for the transformer. Set this parameter based on the corresponding model.
- **guidance_scale_2**: no classifier guidance for transformer_2 of Wan2.2. Set this parameter based on the corresponding model.

The parameters are the same as those of **infer_wan_14b_t2v.sh**. For details, see **Step 6: Performing Inference Using the Wan2.1 Text-to-Video Model**.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

After the inference task is complete, the generated video file **output.mp4** is stored in the **save_path** directory, and the script is stored in the **/home/ma-user/ascendx_video/scripts** directory by default. From there, view the inference result.

Step 10: Performing Inference Using the Wan2.2 Image-to-Video Model

Before starting inference using the image-to-video model, download the sample image and save it to the **/home/ma-user/ascendx_video/scripts** directory.

Figure 4-2 Example



The following scripts are stored in the **/home/ma-user/ascendx_video/scripts/** directory:

- **infer_wan2.2_14b_i2v_480p.sh**: 480P inference script of the Wan image-to-video model Wan2.2-I2V-A14B-Diffusers.
- **infer_wan2.2_14b_i2v_720p.sh**: 720P inference script of the Wan image-to-video model Wan2.2-I2V-A14B-Diffusers.

Run the commands below to start the inference task. The following uses **infer_wan2.2_14b_i2v_480p.sh** as an example.

```
cd /home/ma-user/ascendx_video/scripts/
bash infer_wan2.2_14b_i2v_480p.sh
```

The following describes the parameters of the text-to-video inference script **infer_wan2.2_14b_i2v_480p.sh**. The parameters of script **infer_wan2.2_14b_i2v_720p.sh** are similar to those of **infer_wan2.2_14b_i2v_480p.sh**.

```
export MASTER_ADDR=127.0.0.1
export MASTER_PORT=29505

export PYTORCH_NPU_ALLOC_CONF=expandable_segments:True
export MEMORY_FRAGMENTATION=1
export COMBINED_ENABLE=1
export TASK_QUEUE_ENABLE=2
export TOKENIZERS_PARALLELISM=false
export ASCEND_RT_VISIBLE_DEVICES=0,1,2,3,4,5,6,7
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
N_NPUS=8
torchrun --nproc_per_node=$N_NPUS --master_addr $MASTER_ADDR --master_port $MASTER_PORT ../
infer.py \
    --model Wan2.2-I2V-A14B \
    --pretrained_model_name_or_path ../weights/Wan-AI/Wan2.2-I2V-A14B-Diffusers \
    --task_type i2v \
    --i2v_image_path ./astronaut.jpg \
    --save_path ./output.mp4 \
    --num_inference_steps 40 \
    --width 832 \
    --height 480 \
    --frames 81 \
    --sp $N_NPUS \
    --fsdp \
    --vae_lightning \
    --atten_a8w8 \
    --matmul_a8w8 \
    --rope_fused \
    --guidance_scale 3.5 \
    --guidance_scale_2 3.5 \
    --seed 42 \
    --prompt "An astronaut hatching from an egg, on the surface of the moon, the darkness and depth
of space realised in the background. High quality, ultrarealistic detail and breath-taking movie-like camera
shot." \
    --negative_prompt "vivid tone, overexposure, static, blurred details, subtitle, style, work, painting,
picture, static, overall gray, worst quality, low quality, JPEG compression residue, ugly, incomplete,
redundant fingers, poorly drawn hands, poorly drawn face, deformed, disfigured, deformed limbs, finger
fusion, static picture, messy background, three legs, many people in the background, walking backward"
```

The parameters are the same as those of **infer_wan_14b_t2v.sh**. For details, see **Step 6: Performing Inference Using the Wan2.1 Text-to-Video Model**.

After the inference task is complete, the generated video file **output.mp4** is stored in the **save_path** directory, and the script is stored in the **/home/ma-user/ascendx_video/scripts** directory by default. From there, view the inference result.

Step 11: Performing Inference Using the Wan2.2 Text-to-Image Model

The following scripts are stored in the **/home/ma-user/ascendx_video/scripts/** directory:

- **infer_wan2.2_14b_t2i_480p.sh**: 480P inference script of the Wan text-to-image model Wan2.2-T2V-A14B-Diffusers.
- **infer_wan2.2_14b_t2i_720p.sh**: 720P inference script of the Wan text-to-image model Wan2.2-T2V-A14B-Diffusers.

Run the commands below to start the inference task. The following uses **infer_wan2.2_14b_t2i_480p.sh** as an example.

```
cd /home/ma-user/ascendx_video/scripts/
bash infer_wan2.2_14b_t2i_480p.sh
```

The following describes the parameters of the text-to-image inference script **infer_wan2.2_14b_t2i_480p.sh**. The parameters of script **infer_wan2.2_14b_t2i_720p.sh** are similar to those of **infer_wan2.2_14b_t2i_480p.sh**.

```
export MASTER_ADDR=127.0.0.1
export MASTER_PORT=29505

export PYTORCH_NPU_ALLOC_CONF=expandable_segments:True
export MEMORY_FRAGMENTATION=1
export COMBINED_ENABLE=1
export TASK_QUEUE_ENABLE=2
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
export TOKENIZERS_PARALLELISM=false
export ASCEND_RT_VISIBLE_DEVICES=0,1
N_NPUS=2
torchrun --nproc_per_node=$N_NPUS --master_addr $MASTER_ADDR --master_port $MASTER_PORT ../
infer.py \
    --model Wan2.2-T2V-A14B \
    --pretrained_model_name_or_path ../weights/Wan-AI/Wan2.2-T2V-A14B-Diffusers \
    --task_type t2i \
    --save_path ./output.png \
    --num_inference_steps 40 \
    --width 832 \
    --height 480 \
    --frames 1 \
    --atten_a8w8 \
    --matmul_a8w8 \
    --rope_fused \
    --guidance_scale 3.0 \
    --guidance_scale_2 4.0 \
    --seed 42 \
    --prompt "An astronaut hatching from an egg, on the surface of the moon, the darkness and depth
of space realised in the background. High quality, ultrarealistic detail and breath-taking movie-like camera
shot." \
    --negative_prompt "vivid tone, overexposure, static, blurred details, subtitle, style, work, painting,
picture, static, overall gray, worst quality, low quality, JPEG compression residue, ugly, incomplete,
redundant fingers, poorly drawn hands, poorly drawn face, deformed, disfigured, deformed limbs, finger
fusion, static picture, messy background, three legs, many people in the background, walking backward"
```

The parameters are the same as those of **infer_wan_14b_t2v.sh**. For details, see **Step 6: Performing Inference Using the Wan2.1 Text-to-Video Model**.

After the inference task is complete, the generated image file **output.png** is stored in the **save_path** directory, and the script is stored in the **/home/ma-user/ascendx_video/scripts** directory by default. From there, view the inference result.

Step 12: Performing Inference Using the HunyuanVideo Text-to-Video Model

The following scripts are stored in the **/home/ma-user/ascendx_video/scripts/** directory:

- **infer_hunyuan_video_13b_t2v_sp.sh**: inference script of the HunyuanVideo text-to-video model HunyuanVideo-T2V-13B.

Run the commands below to start the inference task. The following uses **infer_hunyuan_video_13b_t2v_sp** as an example:

```
cd /home/ma-user/ascendx_video/scripts/
bash infer_hunyuan_video_13b_t2v_sp.sh
```

The following describes the parameters of the text-to-video inference script **infer_hunyuan_video_13b_t2v_sp.sh**.

```
export MASTER_ADDR=127.0.0.1
export MASTER_PORT=29502

export PYTORCH_NPU_ALLOC_CONF=expandable_segments:True
export MEMORY_FRAGMENTATION=1
export COMBINED_ENABLE=1
export TASK_QUEUE_ENABLE=2
export TOKENIZERS_PARALLELISM=false
export ASCEND_RT_VISIBLE_DEVICES=0,1,2,3,4,5,6,7
N_NPUS=8
torchrun --nproc_per_node=$N_NPUS --master_addr $MASTER_ADDR --master_port $MASTER_PORT ../
infer.py \
    --model HunyuanVideo-T2V-13B \
    --pretrained_model_name_or_path "../weights/HunyuanVideo" \
    --task_type t2v \
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
    --save_path ./output.mp4 \
    --width 544 \
    --height 960 \
    --frames 129 \
    --sp 8 \
    --guidance_scale 6 \
    --num_inference_steps 50 \
    --seed 42 \
    --fsdp \
    --vae_lightning \
    --turbo_mode next_faiz \
    --prompt "An astronaut floating in space, but instead of stars, there are massive, glowing jellyfish drifting through the void. Their tentacles ripple with shifting, colorful lights, creating a mesmerizing display in the darkness of space. The astronaut moves between them, touching their delicate, translucent bodies, which react to touch with bursts of light. The scene feels dreamlike, as if the astronaut has entered a cosmic undersea world." \
    --negative_prompt "Bright tones, overexposed, static, blurred details, subtitles, style, works, paintings, images, static, overall gray, worst quality, low quality, JPEG compression residue, ugly, incomplete, extra fingers, poorly drawn hands, poorly drawn faces, deformed, disfigured, misshapen limbs, fused fingers, still picture, messy background, three legs, many people in the background, walking backwards"
```

The parameters are the same as those of **infer_wan_14b_t2v.sh**. For details, see **Step 6: Performing Inference Using the Wan2.1 Text-to-Video Model**.

After the inference task is complete, the generated video file **output.mp4** is stored in the **save_path** directory, and the script is stored in the **/home/ma-user/ascendx_video/scripts** directory by default. From there, view the inference result.

Step 13: Performing Inference Using the Cogvideo Text-to-Video Model

The following scripts are stored in the **/home/ma-user/ascendx_video/scripts/** directory:

- **infer_cogvideo_5b.sh**: inference script of the Cogvideo text-to-video model CogVideoX-5b.

Run the commands below to start the inference task. The following uses **infer_cogvideo_5b.sh** as an example:

```
cd /home/ma-user/ascendx_video/scripts/
bash infer_cogvideo_5b.sh
```

The following describes the parameters of the text-to-video inference script **infer_cogvideo_5b.sh**.

```
export MASTER_ADDR=127.0.0.1
export MASTER_PORT=29501

export PYTORCH_NPU_ALLOC_CONF=expandable_segments:True
export MEMORY_FRAGMENTATION=1
export COMBINED_ENABLE=1
export TASK_QUEUE_ENABLE=2
export TOKENIZERS_PARALLELISM=false
export ASCEND_RT_VISIBLE_DEVICES=0,1,2,3,4,5,6,7
N_NPUS=8
torchrun --nproc_per_node=$N_NPUS --master_addr $MASTER_ADDR --master_port $MASTER_PORT ../infer.py \
    --model CogVideoX-5b \
    --pretrained_model_name_or_path ../weights/CogVideoX-5b \
    --save_path ./output.mp4 \
    --width 832 \
    --height 480 \
    --frames 49 \
    --num_inference_steps 50 \
    --sp 8 \
    --vae_lightning \
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
    --turbo_mode next_faiz \
    --fsdp \
    --atten_a8w8 \
    --matmul_a8w8 \
    --prompt "A cat and a dog baking a cake together in a kitchen. The cat is carefully measuring flour, while the dog is stirring the batter with a wooden spoon. The kitchen is cozy, with sunlight streaming through the window." \
    --negative_prompt "Bright tones, overexposed, static, blurred details, subtitles, style, works, paintings, images, static, overall gray, worst quality, low quality, JPEG compression residue, ugly, incomplete, extra fingers, poorly drawn hands, poorly drawn faces, deformed, disfigured, misshapen limbs, fused fingers, still picture, messy background, three legs, many people in the background, walking backwards"
```

The parameters are the same as those of **infer_wan_14b_t2v.sh**. For details, see **Step 6: Performing Inference Using the Wan2.1 Text-to-Video Model**.

After the inference task is complete, the generated video file **output.mp4** is stored in the **save_path** directory, and the script is stored in the **/home/ma-user/ascendx_video/scripts** directory by default. From there, view the inference result.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

5 Migrating Training and Inference from GPU to Ascend

5.1 GPU-to-NPU Migration: Handwritten Digit Recognition

Scenario

In the context of training and inference for AI foundation models, Ascend NPUs, a Chinese-developed computing platform, deliver performance comparable to NVIDIA GPUs. However, developers using existing PyTorch code built for GPUs often worry about the high migration costs and the complexity of adaptation when switching to this platform. The torch_npu package, an adaptation layer for Ascend NPUs, allows standard PyTorch code to run on NPU hardware with minimal modifications, significantly reducing the migration barrier. This tutorial is designed for developers who need to port existing GPU-based PyTorch training code to the Ascend NPU environment.

This tutorial uses MNIST handwritten digit recognition as an example to demonstrate how to migrate PyTorch training code from GPUs to Ascend NPUs. By highlighting the differences between GPU and NPU implementations, this tutorial helps developers quickly understand the migration process and complete training and inference verification on the ModelArts platform.

Concepts

Concept	Description
Ascend NPU	Huawei-developed AI processor that delivers computing performance comparable to NVIDIA GPUs. Built on the Arm architecture, it supports both model training and inference workloads.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Concept	Description
Compute Architecture for Neural Networks (CANN)	Ascend computing architecture, which serves as the foundational software platform for Ascend NPUs. Similar to NVIDIA CUDA, it provides essential capabilities including operator libraries, runtime support, and compilation optimization.
torch_npu	Ascend PyTorch adaptation package, which establishes an adaptation layer between the PyTorch framework and CANN, enabling standard PyTorch code to run on NPUs with minimal modifications.
Compute Unified Device Architecture (CUDA)	NVIDIA's parallel computing platform and programming model, which serves as the foundation for CUDA-related APIs (such as torch.cuda) commonly used in GPU-based training code.

Migration Principles

torch_npu works by creating an adaptation layer between the PyTorch framework and the Ascend CANN software stack. This layer automatically translates PyTorch's CUDA calls into equivalent operations on the Ascend NPU. As a result, developers can migrate models without rewriting the core logic; they only need to replace device-specific API calls to complete the migration.

Typical migration scenario: If your service needs to migrate from an NVIDIA GPU cluster to an Ascend NPU cluster, such as for localization or cost optimization, you can use this guide to quickly adapt your existing PyTorch training and inference code to the NPU environment.

Table 5-1 Mapping between GPU and NPU APIs

Item	GPU	NPU
Device availability	torch.cuda.is_available()	torch.npu.is_available()
Moving tensors to a device	.cuda()	.npu()
Specifying a device	torch.device("cuda")	torch.device("npu")
Loading model mapping	map_location="cuda:0"	map_location="npu:0"
Device monitoring	nvidia-smi	npu-smi info

Expected result: The MNIST handwritten digit recognition model is trained in both the GPU and NPU environments. The code differences before and after the migration are compared to verify that the training accuracy on NPUs is basically aligned with that on GPUs. After the training is complete, the **mnist_gpu.pt** and **mnist_npu.pt** model weight files are generated.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Prerequisites

- You have registered a Huawei Cloud account, completed real-name authentication, and been granted related permissions. For details, see **1. Preparations: Accounts and Permissions**.
- The notebook instance is in the Running state and meets the following requirements. For details about how to create a notebook instance, see **Creating a Notebook Instance**.

  The following table describes the instance specifications, images, and disk specifications.

Parameter	NPU Environment	GPU Environment	Description
Instance Specifications	You are advised to use a single PU (Ascend: 1*ascend-snt9b1) or a higher configuration.	You are advised to use a single-node single-PU flavor or a higher configuration.	The GPU is optional and is used for cross-validation.
Image	Select the preset ModelArts image pytorch_2.7.1-cann_8.3.rc1-py_3.11-hce_2.0.2509-aarch64-snt9b.	Select the preset ModelArts image pytorch_1.12.1-cuda_10.2-py_3.9.11-ubuntu_20.04-x86_64.	Select an image of the corresponding architecture.
Disk Specifications	5 GiB or larger is recommended.	5 GiB or larger is recommended.	The disk is used to store datasets and model weights.

Billing

Running notebook instances in ModelArts development environments use compute and storage resources, which are billed. For details about the billing, see **Development Environment Billing Item**.

Precautions

Before the migration, back up the original GPU code and model weight file. Architecture differences (precaution 1) will make x86 images completely unavailable on NPUs. Before creating an instance, ensure that the image architecture matches the target architecture to avoid resource waste.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Item	Description	Impact Level
Architecture differences	The Ascend NPU runs the Arm system. The Docker image built on the x86 system cannot be directly used. You need to delete the image and rebuild an AArch64 image.	High
Import sequence	**import torch_npu** must be executed after **import torch** and before the NPU functions are used.	High
Operator compatibility	A few custom CUDA operators need to be rewritten using Ascend C. Generally, standard PyTorch operators do not need to be modified.	Medium
Universal model weight	The model weight saved in the **torch.save** file can be used in both GPU and NPU environments. Only the value of **map_location** needs to be adjusted based on the actual architecture.	Low

Core Concepts

The core of this case is a closed-loop process: "environment verification → code migration → training verification → inference comparison." This process provides a clear and intuitive understanding of the entire process of migrating PyTorch code from GPUs to NPUs.

Core of migration: Only three modifications are required (see Step 3).

- 1. Add **import torch_npu** after **import torch**.
- 2. Replace the device identifier **torch.cuda** with **torch.npu**.
- 3. Replace the device identifier **cuda** with **npu**. The code for defining the network model, training logic, and evaluation logic does not need to be modified.

Universal weight: The model weight file (.pt) saved by PyTorch is universal between GPUs and NPUs. You only need to adjust the **map_location** parameter to load the file across devices.

Monitoring comparison: Use **npu-smi info** for NPUs to replace **nvidia-smi** of GPUs. **AICore(%)** corresponds to the NPU usage, **GPU-Util** corresponds to the GPU usage, and **HBM-Usage(MB)** corresponds to the memory usage.

Step 1: Access a Notebook Instance

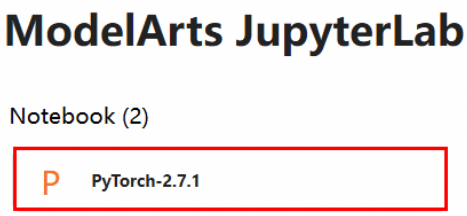
1. Log in to the **ModelArts console** and perform the following operations as required:

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

- New console: Choose **Model Build** > **Notebook**.
- Old console: Choose **Development Space** > **Notebook**.

2. In the **Operation** column of the target notebook instance, click **Access Environment**. In the **Access Method** dialog box, click **Access** on the right of **JupyterLab Access** in the **WebIDE** tab.
3. Go to the **JupyterLab** page. The **ModelArts Launcher** page is automatically displayed. In the **Notebook** area of the **ModelArts Launcher** page, click **PyTorch** to create a PyTorch notebook instance.

Figure 5-1 Opening PyTorch



Step 2: Verify the Environment

Before starting the migration, verify that the current environment is available. After confirming that the device type and computing capability are normal, proceed with the subsequent code migration.

1. Check the device.

   Run the following commands to automatically check whether the current environment uses GPUs, NPUs, or CPUs:

```
import torch

if torch.cuda.is_available():
    device = torch.device("cuda")
    print(f"Current environment: GPU - {torch.cuda.get_device_name(0)}")
elif hasattr(torch, 'npu') and torch.npu.is_available():
    import torch_npu
    device = torch.device("npu")
    print(f"Current environment: NPU - {torch.npu.get_device_name(0)}")
else:
    device = torch.device("cpu")
    print("Current environment: CPU")

print(f"Device identifier: {device}")
```

   After the code is executed, the current device type and device identifier are displayed. The following is an example. If "Current environment: CPU" is displayed, check whether the image is correctly selected or whether the resource specifications are successfully allocated.

   - Output example - NPU:

```
Current environment: NPU - xxx
Device identifier: npu
```

   - Output example - GPU:

```
Current environment: GPU - xxx
Device identifier: cuda
```

2. Verify the matrix operation.

   Verify the device computing capability through matrix multiplication. The following examples show GPU and NPU environments, respectively. Execute them based on the current operating environment.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

- **GPU matrix operation verification:**

```
import torch
try:
    _has_gpu = torch.cuda.is_available()
except Exception:
    _has_gpu = False
if _has_gpu:
    print("===== GPU matrix operation =====")
    x = torch.randn(10000, 10000).cuda()
    y = torch.randn(10000, 10000).cuda()
    z = x.mm(y)
    print(f"Result shape: {z.shape}")
     print(f"Device location: {z.device}")
     print("GPU matrix operation verification passed!")
else:
    print("No GPU is available in the current environment. GPU matrix operation verification is skipped.")
```

Expected output:

```
===== GPU matrix operation =====
Result shape: torch.Size([10000, 10000])
Device location: cuda:0
GPU matrix operation verification passed!
```

- **NPU matrix operation verification:**

```
import torch
try:
    import torch_npu
    _has_npu = torch.npu.is_available()
except Exception:
    _has_npu = False
if _has_npu:
    print("===== NPU matrix operation =====")
    x = torch.randn(10000, 10000).npu()
    y = torch.randn(10000, 10000).npu()
    z = x.mm(y)
    print(f"Result shape: {z.shape}")
     print(f"Device location: {z.device}")
    print("NPU matrix operation verification passed!")
else:
    print("No NPU is available in the current environment. NPU matrix operation verification is skipped.")
```

Expected output:

```
===== NPU matrix operation =====
Result shape: torch.Size([10000, 10000])
Device location: npu:0
NPU matrix operation verification passed!
```

Key points in migration: In the NPU environment, you only need to add **import torch_npu** and replace **.cuda ()** with **.npu ()**. The rest of the code remains unchanged.

Verification method: If the device is available, the message "Matrix operation verification passed!" and the result shape torch.Size([10000, 10000]) should be displayed.

Step 3: Perform MNIST Handwritten Digit Recognition

This step completes the entire training process for MNIST handwritten digit recognition, including defining the network model, preparing and loading data, defining training and evaluation functions, and performing GPU/NPU-based training. The only difference lies in the training code, which varies between GPUs and NPUs. All other code remains consistent across both environments.

1. Define a network model.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

The definition of the network model is hardware-agnostic, meaning the abstract definition remains consistent across different devices (GPUs or NPUs). This case employs a LeNet-5 variant convolutional neural network designed to classify 28 x 28 grayscale images. The architecture comprises two convolutional layers (Conv2d), two Dropout layers, and two fully connected layers (Linear), ultimately producing a probability distribution across 10 classes.

```
import torch
import torch.nn as nn
import torch.nn.functional as F

class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, 3, 1)
        self.conv2 = nn.Conv2d(32, 64, 3, 1)
        self.dropout1 = nn.Dropout(0.25)
        self.dropout2 = nn.Dropout(0.5)
        self.fc1 = nn.Linear(9216, 128)
        self.fc2 = nn.Linear(128, 10)

    def forward(self, x):
        x = self.conv1(x)
        x = F.relu(x)
        x = self.conv2(x)
        x = F.relu(x)
        x = F.max_pool2d(x, 2)
        x = self.dropout1(x)
        x = torch.flatten(x, 1)
        x = self.fc1(x)
        x = F.relu(x)
        x = self.dropout2(x)
        x = self.fc2(x)
        output = F.log_softmax(x, dim=1)
        return output

print("Network model defined.")
```

Expected output:

```
Network model defined.
```

2. Prepare data.

   Download the MNIST dataset and build a data loader. The code in the GPU and NPU environments is the same. The MNIST dataset can be downloaded in either of the following ways. Select a method based on your network environment.

Download Method	Application Scenario	Description
Method 1: Directly download the dataset.	The network can access GitHub.	Download the raw data file from GitHub.
Method 2: Download the dataset from an OBS bucket.	The network is restricted.	Download the packaged data from the Huawei Cloud OBS bucket.

   - **Method 1: Directly download the dataset.**

     ```
     # Standard mode
     !mkdir -p ./data/MNIST/raw
     !wget https://raw.githubusercontent.com/leo-mao/MNIST_data/master/train-images-idx3-
     ```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
ubyte.gz -O ./data/MNIST/raw/train-images-idx3-ubyte.gz
!wget https://raw.githubusercontent.com/leo-mao/MNIST_data/master/train-labels-idx1-
ubyte.gz -O ./data/MNIST/raw/train-labels-idx1-ubyte.gz
!wget https://raw.githubusercontent.com/leo-mao/MNIST_data/master/t10k-images-idx3-
ubyte.gz -O ./data/MNIST/raw/t10k-images-idx3-ubyte.gz
!wget https://raw.githubusercontent.com/leo-mao/MNIST_data/master/t10k-labels-idx1-
ubyte.gz -O ./data/MNIST/raw/t10k-labels-idx1-ubyte.gz
import os
import gzip
from torchvision import datasets, transforms
os.makedirs('./data/MNIST/raw', exist_ok=True)

# Decompress the .gz file.
for f in os.listdir('./data/MNIST/raw/'):
    if f.endswith('.gz'):
        with gzip.open(f'./data/MNIST/raw/{f}', 'rb') as gz:
            with open(f'./data/MNIST/raw/{f[:-3]}', 'wb') as out:
                out.write(gz.read())
```

- **Method 2: Download the dataset from an OBS bucket.**

```
# Download the dataset from the OBS bucket.
!pip install download

%env no_proxy='a.test.com,127.0.0.1,2.2.2.2'
from download import download
# Download the MNIST dataset.
url = "https://eduhicomputing.obs.cn-south-1.myhuaweicloud.com:443/%20ComputerVision2/
MNIST_Data.zip"
path = download(url, "./data/MNIST/raw", kind="zip", replace=True)
!mv ./data/MNIST/raw/MNIST_Data/test/* ./data/MNIST/raw/MNIST_Data/train/* ./data/
MNIST/raw/ && rm -rf ./data/MNIST/raw/MNIST_Data
```

3. Load the dataset. Use torchvision to load the MNIST dataset and create the DataLoader for the training and test sets.

```
from torchvision import datasets, transforms
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.1307,), (0.3081,))
])
dataset1 = datasets.MNIST('./data', train=True, download=False, transform=transform)
dataset2 = datasets.MNIST('./data', train=False, download=False, transform=transform)
print(f"Training set size: {len(dataset1)}")
print(f"Test set size: {len(dataset2)}")
```

   Expected output:

```
Training set size: 60000
Test set size: 10000
```

   If the data fails to be loaded, check whether the data file is completely downloaded.

4. Define training and evaluation functions. Define the training function **train** and evaluation function **test**. The functions automatically adapt to GPUs or NPUs based on the **device** parameter in the request. The function logic does not need to be modified.

   The training function moves data and tags to the specified device in each batch, and the evaluation function calculates the loss and accuracy. The two functions use the **device** parameter to adapt to different devices and do not need to be modified.

```
def train(model, device, train_loader, optimizer, epoch):
    """Train the model and return the average loss and accuracy."""
    model.train()
    total_loss = 0
    correct = 0
    total = 0
    for batch_idx, (data, target) in enumerate(train_loader):
        data, target = data.to(device), target.to(device)
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
        optimizer.zero_grad()
        output = model(data)
        loss = F.nll_loss(output, target)
        loss.backward()
        optimizer.step()
        total_loss += loss.item()
        pred = output.argmax(dim=1, keepdim=True)
        correct += pred.eq(target.view_as(pred)).sum().item()
        total += len(data)
    avg_loss = total_loss / len(train_loader)
    accuracy = 100. * correct / total
    print(f'Train Epoch: {epoch} | Loss: {avg_loss:.4f} | Accuracy: {correct}/{total} ({accuracy:.0f}%)')
    return avg_loss, accuracy


def test(model, device, test_loader):
    """Evaluate the model and return the average loss and accuracy."""
    model.eval()
    test_loss = 0
    correct = 0
    with torch.no_grad():
        for data, target in test_loader:
            data, target = data.to(device), target.to(device)
            output = model(data)
            test_loss += F.nll_loss(output, target, reduction='sum').item()
            pred = output.argmax(dim=1, keepdim=True)
            correct += pred.eq(target.view_as(pred)).sum().item()

    test_loss /= len(test_loader.dataset)
    accuracy = 100. * correct / len(test_loader.dataset)
    print(f'Test  | Average loss: {test_loss:.4f} | Accuracy: {correct}/{len(test_loader.dataset)}
({accuracy:.0f}%)')
    return test_loss, accuracy

print("Training and evaluation functions defined.")
```

Expected output:

```
Training and evaluation functions defined.
```

5. Perform GPU/NPU-based training. Use GPUs and NPUs to train the model and compare the code before and after the migration. Save the model weight file after training for three epochs.
   - **GPU-based training**: The following code runs on a GPU-based instance.

     ```
     import torch
     import torch.optim as optim
     from torch.optim.lr_scheduler import StepLR
     try:
         import torch.cuda
         cuda_available = torch.cuda.is_available()
     except (ImportError, AttributeError):
         cuda_available = False

     # ====== GPU-based training ======
     if cuda_available:
         device = torch.device("cuda")
         print(f"Device used: {device} - {torch.cuda.get_device_name(0)}")
         train_kwargs = {'batch_size': 64, 'shuffle': True}
         test_kwargs = {'batch_size': 1000, 'shuffle': False}
         train_loader = torch.utils.data.DataLoader(dataset1, **train_kwargs)
         test_loader = torch.utils.data.DataLoader(dataset2, **test_kwargs)
         model = Net().to(device)
         optimizer = optim.Adadelta(model.parameters(), lr=1.0)
         scheduler = StepLR(optimizer, step_size=1, gamma=0.7)
         for epoch in range(1, 4):
             train(model, device, train_loader, optimizer, epoch)
             test(model, device, test_loader)
             scheduler.step()
         torch.save(model.state_dict(), "mnist_gpu.pt")
         print("GPU-based training completed. The model has been saved as mnist_gpu.pt.")
     ```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
else:
    print("No GPU is available in the current environment. GPU-based training is skipped.")
```

Expected output:

```
Devices used: cuda - Tesla V100S-PCIE-32GB
Train Epoch: 1 | Loss: 0.1968 | Accuracy: 56419/60000 (94%)
Test  | Average loss: 0.0542 | Accuracy: 9826/10000 (98%)
Train Epoch: 2 | Loss: 0.0759 | Accuracy: 58647/60000 (98%)
Test  | Average loss: 0.0417 | Accuracy: 9867/10000 (99%)
Train Epoch: 3 | Loss: 0.0580 | Accuracy: 59008/60000 (98%)
Test  | Average loss: 0.0313 | Accuracy: 9891/10000 (99%)
GPU-based training completed. The model has been saved as mnist_gpu.pt.
```

Verification method: If GPU image resources are used, the message "GPU-based training completed. The model has been saved as mnist_gpu.pt." should be displayed after training. The test accuracy for all three epochs should be within the range of 97% to 99%. If the message "No GPU available in the current environment." is displayed, check whether the instance specifications include GPUs.

- **NPU-based training**: The following code runs on an NPU-based instance.

  Key points for migration: Compared with **GPU-based training**, only three changes are made:

  i. Add **import torch_npu**.

  ii. torch.cuda → torch.npu

  iii. cuda → npu

```
import torch
import torch.optim as optim
from torch.optim.lr_scheduler import StepLR
try:
    import torch_npu # [Change 1] Import the NPU adaptation package.
    npu_available = torch.npu.is_available() # [Change 2] torch.cuda → torch.npu
except (ImportError, AttributeError):
    npu_available = False

# ====== NPU-based training ======
if npu_available:
    device = torch.device("npu")   # [Change 3] cuda → npu
    print(f"Device used: {device} - {torch.npu.get_device_name(0)}")

    train_kwargs = {'batch_size': 64, 'shuffle': True}
    test_kwargs = {'batch_size': 1000, 'shuffle': False}

    train_loader = torch.utils.data.DataLoader(dataset1, **train_kwargs)
    test_loader = torch.utils.data.DataLoader(dataset2, **test_kwargs)

    model = Net().to(device)
    optimizer = optim.Adadelta(model.parameters(), lr=1.0)
    scheduler = StepLR(optimizer, step_size=1, gamma=0.7)

    for epoch in range(1, 4):
        train(model, device, train_loader, optimizer, epoch)
        test(model, device, test_loader)
        scheduler.step()

    torch.save(model.state_dict(), "mnist_npu.pt")
    print("NPU-based training completed. The model has been saved as mnist_npu.pt.")
else:
    print("No NPU is available in the current environment. NPU-based training is skipped.")
```

Expected output:

```
Device used: npu - Ascend910B4
Train Epoch: 1 | Loss: 0.2067 | Accuracy: 56290/60000 (94%)
Test  | Average loss: 0.0442 | Accuracy: 9844/10000 (98%)
Train Epoch: 2 | Loss: 0.0761 | Accuracy: 58734/60000 (98%)
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
Test  | Average loss: 0.0344 | Accuracy: 9885/10000 (99%)
Train Epoch: 3 | Loss: 0.0544 | Accuracy: 59053/60000 (98%)
Test  | Average loss: 0.0319 | Accuracy: 9890/10000 (99%)
NPU-based training completed. The model has been saved as mnist_npu.pt.
```

Verification method: If NPU image specifications are used, the message "NPU-based training completed. The model has been saved as mnist_npu.pt." should be displayed after training. The test accuracy for all three epochs should be basically the same as that for GPU-based training. If the message "No NPU available in the current environment." is displayed, check whether the CANN version is selected for the image.

Core Diff Comparison

```
 + import torch_npu                # [Change 1]: Import the NPU adaptation package.
- if torch.cuda.is_available():    # [Change 2]: Check the device.
+ if torch.npu.is_available():
- device = torch.device("cuda")      # [Change 3]: Specify the device.
+ device = torch.device("npu")
```

Step 4: Perform Model Inference and Cross-Device Loading

In this step, the trained model is used for inference and verification, including loading the model weight file, predicting images in the test set, and visualizing the results. The PyTorch weight file is compatible between GPUs and NPUs. You only need to adjust the **map_location** parameter to load the file across devices.

1. Load the trained model weight file for inference. The code automatically detects the current device type and loads the corresponding model weight.

   The weight file (.pt) saved by PyTorch is compatible between GPUs and NPUs. You only need to adjust the **map_location** parameter.

   Key points for migration: The model weight file in this case is universal. In the NPU scenario, only **map_location** needs to be changed from **cuda:0** to **npu:0**. In addition, the code for network definition, training logic, evaluation logic, and data loading is the same.

   ```
   import torch

   # Automatically detect the device and load the corresponding model.
   if torch.cuda.is_available():
       device = torch.device("cuda")
       model_path = "mnist_gpu.pt"
       map_location = "cuda:0"
   elif hasattr(torch, 'npu') and torch.npu.is_available():
       import torch_npu
       device = torch.device("npu")
       model_path = "mnist_npu.pt"
       map_location = "npu:0" # In the NPU scenario, change map_location from "cuda:0" to "npu:0".
   else:
       device = torch.device("cpu")
       model_path = "mnist_gpu.pt"
       map_location = "cpu"

   model = Net().to(device)
   model.load_state_dict(torch.load(model_path, map_location=map_location))
   model.eval()
   print(f"Model loaded. Device: {device}")
   ```

   Expected output:

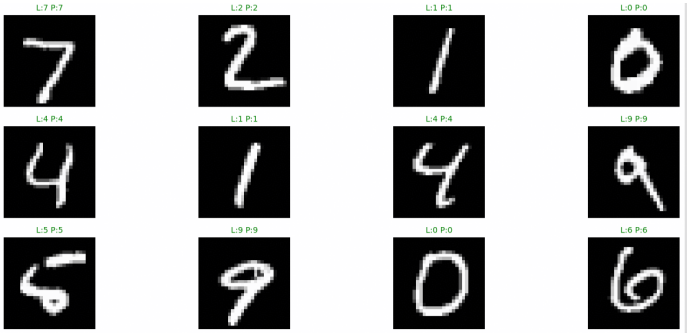
   - NPU:
     ```
     Model loaded. Device: npu
     ```
   - GPU:
     ```
     Model loaded. Device: cuda
     ```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

2. Select 12 images from the test dataset, use the loaded model to perform inference, and display the comparison between the prediction results and the actual tags.

```
import matplotlib.pyplot as plt
num = 12
cols = 4
rows = (num + cols - 1) // cols
fig, axes = plt.subplots(rows, cols, figsize=(15, 6))
model.eval()
with torch.no_grad():
    for i in range(num):
        image, label = dataset2[i]
        output = model(image.unsqueeze(0).to(device))
        pred = output.argmax(dim=1).item()
        ax = axes[i // cols, i % cols]
        ax.imshow(image.squeeze(), cmap='gray')
        color = 'green' if pred == label else 'red'
        ax.set_title(f'L:{label} P:{pred}', color=color, fontsize=10)
        ax.axis('off')
plt.tight_layout()
plt.show()
```

Expected output:



In the prediction result graph, the green title indicates a correct prediction, and the red title indicates an incorrect prediction.

Step 5: Monitor the Device

During training and inference, you can run device monitoring commands to check the compute usage and memory usage of NPUs or GPUs. This helps you determine whether the training is running properly. For GPUs, run the **nvidia-smi** command. For NPUs, run the **npu-smi info** command. The following table lists the core monitoring metrics of the two types of devices.

- **GPU monitoring**

  Run the **nvidia-smi** command on the GPU instance to check the GPU usage and memory usage.

```
import torch

if torch.cuda.is_available():
    !nvidia-smi
else:
    print("No GPU is available in the current environment. nvidia-smi is skipped.")
```

  Expected output:

```
Wed Jul 29 11:50:13 2026
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.57.02    Driver Version: 470.57.02    CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  xxx...  On   | 00000000:00:0D.0 Off |                    0 |
| N/A   35C    P0    37W / 250W |   2925MiB / 32510MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
```

- **NPU monitoring**

  Run the **npu-smi info** command on the NPU instance to check the NPU usage and memory usage. **AICore(%)** indicates the compute usage of NPUs, which is equivalent to **GPU-Util** of GPUs. **HBM-Usage(MB)** indicates the memory usage, which is equivalent to **Memory-Usage** of GPUs.

  ```
  import torch

  if hasattr(torch, 'npu') and torch.npu.is_available():
      !npu-smi info
  else:
      print("No NPU is available in the current environment. npu-smi info is skipped.")
  ```

  Expected output:

  ```
  +------------------------------------------------------------------------------------------------+
  | npu-smi 25.5.1                  Version: 25.5.1                                                |
  +---------------------------+---------------+----------------------------------------------------+
  | NPU   Name                | Health        | Power(W)     Temp(C)           Hugepages-Usage(page)|
  | Chip                      | Bus-Id        | AICore(%)    Memory-Usage(MB)  HBM-Usage(MB)        |
  +===========================+===============+====================================================+
  | 0     xxx                 | OK            | 87.5         42                0    / 0             |
  | 0                         | 0000:C1:00.0  | 0            0    / 0          4822 / 32768         |
  +===========================+===============+====================================================+
  +---------------------------+---------------+----------------------------------------------------+
  | NPU     Chip              | Process id    | Process name             | Process memory(MB)      |
  +===========================+===============+====================================================+
  | 0       0                 | 3180          | python                   | 1999                    |
  +===========================+===============+====================================================+
  ```

Table 5-2 Comparison between GPU and NPU monitoring metrics

Metric	GPU (nvidia-smi)	NPU (npu-smi info)
Compute usage	GPU-Util	AICore(%)
Memory usage	Memory-Usage	HBM-Usage(MB)
Device temperature	Temp	Temp(C)
Power consumption	Pwr	Power(W)

Real-time monitoring (continuous refresh)

If continuous monitoring is required, run the following command:

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

- GPU:
  ```
  watch -n 1 nvidia-smi (refreshed every 1 second)
  ```
- NPU:
  ```
  watch -n 1 npu-smi info (refreshed every 1 second)
  ```

FAQ

- What do I do if a message is displayed indicating that no NPU is available in the current environment when I run the NPU training code?

  Check the following items in sequence:

  a. Whether the NPU type is selected for the resource specifications of the notebook instance.

  b. Whether the AArch64 image that contains CANN is selected.

  c. Whether the instance has been started and NPU resources have been allocated.

- Can the model weights saved during GPU-based training be directly loaded on NPUs?

  Yes. The PyTorch weight file is device-agnostic. You only need to adjust the **map_location** parameter. In some scenarios, you need to load the weights into CPUs by setting **map_location=cpu** and then convert them to NPUs.

Helpful Links

Document Name	Description
torch_npu official documentation	API reference and usage description of the Ascend PyTorch adaptation package
ModelArts Notebook User Guide	Methods for creating, configuring, and managing notebook instances
ModelArts Billing	Billing standards and fee estimation for resource specifications

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

6 Notebook

6.1 Migrating the Conda Environment on a Notebook Instance to an SFS Disk

This section describes how to migrate the Conda environment on a notebook instance to an SFS disk. In this way, the Conda environment will not be lost after the notebook instance is restarted.

The procedure is as follows:

1. **Creating a Virtual Environment and Saving It to the SFS Directory**
2. **Cloning the Existing Virtual Environments to the SFS Disk**
3. **Restarting the Image to Activate the Virtual Environment in the SFS Disk**
4. **Saving and Sharing the Virtual Environment**

Prerequisites

You have created a notebook instance by setting **Resource Pool Type** to **Dedicated resource pool** and **Data Disk Type** to **Scalable File Service Turbo (SFS Turbo)** and opened the terminal. For details, see **Creating a Notebook Instance**.

Creating a Virtual Environment and Saving It to the SFS Directory

Create a conda virtual environment.

```
# shell
conda create --prefix /home/ma-user/work/envs/user_conda/sfs-new-env python=3.7.10 -y
```

View the existing conda virtual environments. The name of the newly created virtual environment may be empty in the output.

```
# shell
conda env list
# conda environments:
#
base                  /home/ma-user/anaconda3
PyTorch-1.8              /home/ma-user/anaconda3/envs/PyTorch-1.8
python-3.7.10         *  /home/ma-user/anaconda3/envs/python-3.7.10
                      /home/ma-user/work/envs/user_conda/sfs-new-env
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Append the new virtual environment to conda envs.

```
# shell
conda config --append envs_dirs /home/ma-user/work/envs/user_conda/
```

View the existing conda virtual environments. The new virtual environment is properly displayed, and you can switch to it by name.

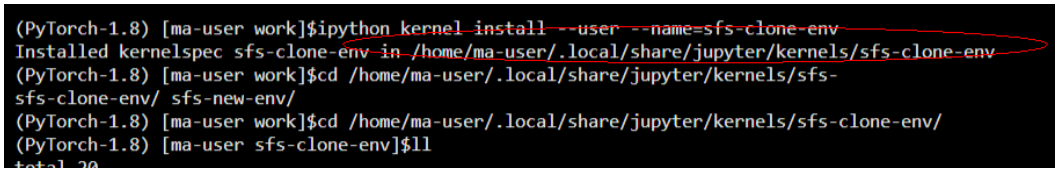
```
# shell
conda env list
conda activate sfs-new-env
# conda environments:
#
base                  /home/ma-user/anaconda3
PyTorch-1.8              /home/ma-user/anaconda3/envs/PyTorch-1.8
python-3.7.10         *  /home/ma-user/anaconda3/envs/python-3.7.10
sfs-new-env              /home/ma-user/work/envs/user_conda/sfs-new-env
```

(Optional) Register the new virtual environment with the JupyterLab kernel, so that you can directly use it in JupyterLab.

```
# shell
pip install ipykernel
ipython kernel install --user --name=sfs-new-env
rm -rf /home/ma-user/.local/share/jupyter/kernels/sfs-new-env/logo-*
```

Note: **.local/share/jupyter/kernels/sfs-new-env** is used as an example only. Replace it with the actual installation path.

Figure 6-1 Installation path output



Refresh the JupyterLab page. The new kernel is displayed.

NOTE

After the notebook instance is restarted, the kernel needs to be registered again.

Cloning the Existing Virtual Environments to the SFS Disk

```
# shell
conda create --prefix /home/ma-user/work/envs/user_conda/sfs-clone-env --clone PyTorch-1.8 -
y
Source:      /home/ma-user/anaconda3/envs/PyTorch-1.8
Destination: /home/ma-user/work/envs/user_conda/sfs-clone-env
Packages: 20
Files: 39687
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate /home/ma-user/work/envs/user_conda/sfs-clone-env
#
# To deactivate an active environment, use
#
#     $ conda deactivate
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

View the cloned virtual environments. If the name of the newly created virtual environment is empty, handle the issue according to **Append the new virtual environment to conda envs**.

```
# shell
conda env list
# conda environments:
#
base                  /home/ma-user/anaconda3
PyTorch-1.8              /home/ma-user/anaconda3/envs/PyTorch-1.8
python-3.7.10             /home/ma-user/anaconda3/envs/python-3.7.10
sfs-clone-env            /home/ma-user/work/envs/user_conda/sfs-clone-env
sfs-new-env           *  /home/ma-user/work/envs/user_conda/sfs-new-env
```

(Optional) Register the new virtual environment with the JupyterLab kernel, so that you can directly use it in JupyterLab.

```
# shell
pip install ipykernel
ipython kernel install --user --name=sfs-clone-env
rm -rf /home/ma-user/.local/share/jupyter/kernels/sfs-clone-env/logo-*
```

Note: **.local/share/jupyter/kernels/sfs-clone-env** is used as an example only. Replace it with the actual installation path.

Refresh the JupyterLab page. The new kernel is displayed.

Restarting the Image to Activate the Virtual Environment in the SFS Disk

Method 1: Use the complete conda env path.

```
# shell
conda activate /home/ma-user/work/envs/user_conda/sfs-new-env
```

Method 2: Append the virtual environment to conda envs and activate it using its name.

```
# shell
conda config --append envs_dirs /home/ma-user/work/envs/user_conda/
conda activate sfs-new-env
```

Method 3: Use Python or pip in the virtual environment.

```
# shell
/home/ma-user/work/envs/user_conda/sfs-new-env/bin/pip list
/home/ma-user/work/envs/user_conda/sfs-new-env/bin/python -V
```

Saving and Sharing the Virtual Environment

Package the virtual environment to be migrated.

```
# shell
pip install conda-pack
conda pack -n sfs-clone-env -o sfs-clone-env.tar.gz --ignore-editable-packages
Collecting packages...
Packing environment at '/home/ma-user/work/envs/user_conda/sfs-clone-env' to 'sfs-clone-env.tar.gz'
[########################################] | 100% Completed |  3min 33.9s
```

Decompress the package to the SFS directory.

```
# shell
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
mkdir /home/ma-user/work/envs/user_conda/sfs-tar-env
tar -zxvf sfs-clone-env.tar.gz -C /home/ma-user/work/envs/user_conda/sfs-tar-env
```

View the existing conda virtual environments.

```
# shell
conda env list
# conda environments:
#
base                  /home/ma-user/anaconda3
PyTorch-1.8           *  /home/ma-user/anaconda3/envs/PyTorch-1.8
python-3.7.10            /home/ma-user/anaconda3/envs/python-3.7.10
sfs-clone-env            /home/ma-user/work/envs/user_conda/sfs-clone-env
sfs-new-env              /home/ma-user/work/envs/user_conda/sfs-new-env
sfs-tar-env              /home/ma-user/work/envs/user_conda/sfs-tar-env
test-env                 /home/ma-user/work/envs/user_conda/test-env
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

7 Model Training

7.1 Running a GPU-based Training Job on ModelArts

7.1.1 Scenarios

The data volume and compute used for training vary among AI models. Select a proper storage and training solution to improve training efficiency and resource cost-effectiveness. ModelArts supports multiple training scenarios to meet different requirements, including single-node single-PU, single-node multi-PU, and multi-node multi-PU.

There are public resource pools and dedicated resource pools. If you use a dedicated resource pool, the resources are not shared with other users, ensuring high efficiency. For enterprises with multiple users, use a dedicated resource pool for AI model training.

This section provides an E2E guidance to help you understand how to select a proper training solution and perform model training on ModelArts.

For different data volumes and algorithms, the solutions are as follows:

- Single-node single-PU: If the data volume is small (1 GB training data) and the compute is low (one Vnt1), use OBS parallel file system to store data and code.
- Single-node multi-PU: If the data volume is medium (50 GB training data) and the compute is medium (eight Vnt1s), use SFS to store data and code.
- Multi-node multi-PU: If the data volume is large (1 TB training data) and the compute is high (four nodes with eight Vnt1s), use SFS to store data and a common OBS bucket to store code, and use distributed training.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Table 7-1 Services required in different scenarios and purchase recommendations

Scenario	OBS	SFS	SWR	DEW	ModelArts	VPC	ECS	EVS
Single-node single-PU	Pay-per-use (parallel file system)	×	Free	Free	Monthly	Free	×	Pay-per-use
Single-node multi-PU	×	Monthly (HPC 500 GB)	Free	Free	Monthly	Free	Monthly (Ubuntu 18.04, at least 2 vCPUs and 8 GB memory, 100 GB local storage space, dynamic BGP with EIP, and 10 Mbit/s bandwidth)	×
Multi-node multi-PU	Pay-per-use (common OBS bucket)	Monthly (HPC 500 GB)	Free	Free	Monthly	Free	Monthly (Ubuntu 18.04, at least 2 vCPUs and 8 GB memory, 100 GB local storage space, dynamic BGP with EIP, and 10 Mbit/s bandwidth)	×

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Table 7-2 Training performance of different open-source datasets

Algorithm and Data	Resource Flavor	Number of Epochs	Estimated Running Duration (hh:mm:ss)
Algorithm: PyTorch official example for ImageNet Data: ImageNet classification data subset	One node with one Vnt1	10	0:05:03
Algorithm: YOLOX Data: COCO 2017 dataset	One node with one Vnt1	10	03:33:13
	One node with eight Vnt1s	10	01:11:48
	Four nodes with eight Vnt1s	10	0:36:17
Algorithm: Swin-Transformer Data: ImageNet21K	One node with one Vnt1	10	197:25:03
	One node with eight Vnt1s	10	26:10:25
	Four nodes with eight Vnt1s	10	07:08:44

Table 7-3 Average response time for different training operations

Operation	Description	Estimated Duration
Downloading an image	Time when an image (25 GB) is downloaded for the first time	8 minutes
Scheduling resources	Duration from the time when a training job starts to be created to the time when the training job becomes running (resources are sufficient and the image is cached)	20 seconds
Accessing the training list page	Time to access the training job list page with 50 records on it	6 seconds
Loading logs	Time to load 1 MB logs on the training details page	2.5 seconds

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Operation	Description	Estimated Duration
Accessing the training details page	Time to access the training details page where there are no logs	2.5 seconds
Accessing the JupyterLab page	Time to access the JupyterLab page and load the page content	0.5 seconds
Accessing the notebook list page	Time to access the notebook list page with 50 instances on it	4.5 seconds

NOTE

The preceding data is for reference only. The image download time depends on the node specifications, disk type (high I/O or common I/O), and whether SSD is used.

7.1.2 Preparations

Before using dedicated resource pools for training on ModelArts, perform the following operations.

Purchasing Service Resources

Table 7-4 Purchasing service resources

Service	Description	Details
SFS Turbo	SFS charges you based on the storage capacity and usage duration you select. You can also buy a yearly or monthly package that suits your resource needs and plans. In case of arrears, you have 15 days to renew the service. Otherwise, your file system resources will be removed. You can use SFS to store data and code.	**How Do I Purchase SFS?**
SWR	SWR has two editions: Enterprise Edition and Shared Edition. SWR Shared Edition is free, metered by storage space and traffic. SWR Enterprise Edition supports pay-per-use billing mode. You can use SWR to upload custom images.	**Uploading an Image**

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Service	Description	Details
OBS	OBS offers two billing modes: pay-per-use and yearly/monthly. Choose the one that suits your needs. OBS has two storage modes. For single-node and single-PU training, use a file system. For multi-node and multi-PU training, use a common OBS bucket.	• Creating a Bucket • Creating a Parallel File System
Virtual Private Cloud (VPC)	A VPC enables you to provision logically isolated, configurable, and manageable virtual networks. VPC interconnection allows you to use resources across VPCs, improving resource utilization.	Creating a VPC
Elastic Cloud Server (ECS)	ECSs are more cost-effective than physical servers. Within minutes, you can obtain ECS resources from the cloud service platform. ECS resources are flexible and on-demand. You can use ECS to mount SFS Turbo storage. **NOTE** The ECS and SFS must be in the same VPC for mounting SFS.	Purchasing an ECS in Custom Config Mode
Data Encryption Workshop (DEW)	When you use a notebook instance for code debugging, if you want to enable remote SSH development, you need DEW to select a key pair. In this way, you can log in to the ECS using the key pair, improving security. Key pairs can be created free of charge.	Creating a Key Pair

Assigning Permissions

Step 1 Configure IAM permissions.

1. Use a Huawei Cloud tenant account to create a developer user group **user_group** and add developer accounts to **user_group**. For details, see **Step 1 Creating a User Group and Adding Users to the User Group**.
2. Create a custom policy.
   a. Log in to the **Huawei Cloud console** as a Huawei Cloud tenant user, hover over your username in the upper right corner, and choose **Identity and Access Management** from the drop-down list to go to the IAM console.
   b. In the navigation pane on the left, choose **Permissions** > **Policies/Roles**. Click **Create Custom Policy** in the upper right corner. On the displayed page, enter **Policy1** or **Policy2** for **Policy Name**, select **JSON** for **Policy View**, configure the policy content, and click **OK**.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

NOTE

To define permissions required to access both global and project-level services, enclose the permissions in two separate custom policies for refined authorization. **Learn more**.

- The content of **Policy1** is as follows:

```
{
   "Version": "1.1",
   "Statement": [
      {
         "Action": [
            "modelarts:*:*"
         ],
         "Effect": "Allow"
      },
      {
         "Action": [
            "modelarts:pool:create",
            "modelarts:pool:update",
            "modelarts:pool:delete"
         ],
         "Effect": "Deny"
      },
      {
         "Action": [
            "sfsturbo:*:*",
            "vpc:*:*",
            "dss:*:get",
            "dss:*:list"
         ],
         "Effect": "Allow"
      },
      {
         "Action": [
            "ecs:*:*",
            "evs:*:get",
            "evs:*:list",
            "evs:volumes:create",
            "evs:volumes:delete",
            "evs:volumes:attach",
            "evs:volumes:detach",
            "evs:volumes:manage",
            "evs:volumes:update",
            "evs:volumes:use",
            "evs:volumes:uploadImage",
            "evs:snapshots:create",
            "vpc:*:get",
            "vpc:*:list",
            "vpc:networks:create",
            "vpc:networks:update",
            "vpc:subnets:update",
            "vpc:subnets:create",
            "vpc:ports:*",
            "vpc:routers:get",
            "vpc:routers:update",
            "vpc:securityGroups:*",
            "vpc:securityGroupRules:*",
            "vpc:floatingIps:*",
            "vpc:publicIps:*",
            "ims:images:create",
            "ims:images:delete",
            "ims:images:get",
            "ims:images:list",
            "ims:images:update",
            "ims:images:upload"
         ],
         "Effect": "Allow"
      },
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
    {
        "Action": [
            "vpc:*:*",
            "ecs:*:get*",
            "ecs:*:list*"
        ],
        "Effect": "Allow"
    },
    {
        "Action": [
            "kms:cmk:*",
            "kms:dek:*",
            "kms:grant:*",
            "kms:cmkTag:*",
            "kms:partition:*"
        ],
        "Effect": "Allow"
    }
  ]
}
```

- The content of **Policy2** is as follows:

```
{
  "Version": "1.1",
  "Statement": [
    {
        "Action": [
            "obs:bucket:ListAllMybuckets",
            "obs:bucket:HeadBucket",
            "obs:bucket:ListBucket",
            "obs:bucket:GetBucketLocation",
            "obs:object:GetObject",
            "obs:object:GetObjectVersion",
            "obs:object:PutObject",
            "obs:object:DeleteObject",
            "obs:object:DeleteObjectVersion",
            "obs:object:ListMultipartUploadParts",
            "obs:object:AbortMultipartUpload",
            "obs:object:GetObjectAcl",
            "obs:object:GetObjectVersionAcl"
        ],
        "Effect": "Allow"
    }
  ]
}
```

3. Grant the custom policy to the developer user group **user_group**.
   a. In the navigation pane of the IAM console, choose **User Groups**. On the **User Groups** page, locate the row that contains **user_group**, click **Authorize** in the **Operation** column, and select **Policy1**, **Policy2**, and **SWR Admin**. Click **Next**.

      NOTE

      SoftWare Repository for Container (SWR) permissions include SWR FullAccess, SWR OperateAccess, and SWR ReadOnlyAccess. However, these permissions are available only for SWR Enterprise Edition, which has been suspended OBT. You need to select **SWR Admin**.

   b. Set the minimum authorization scope, select **All resources** for **Scope** and click **OK**.

For more information about permissions management, see **Basic Concepts of ModelArts Permissions Management**.

Step 2 Configure ModelArts agency permissions.

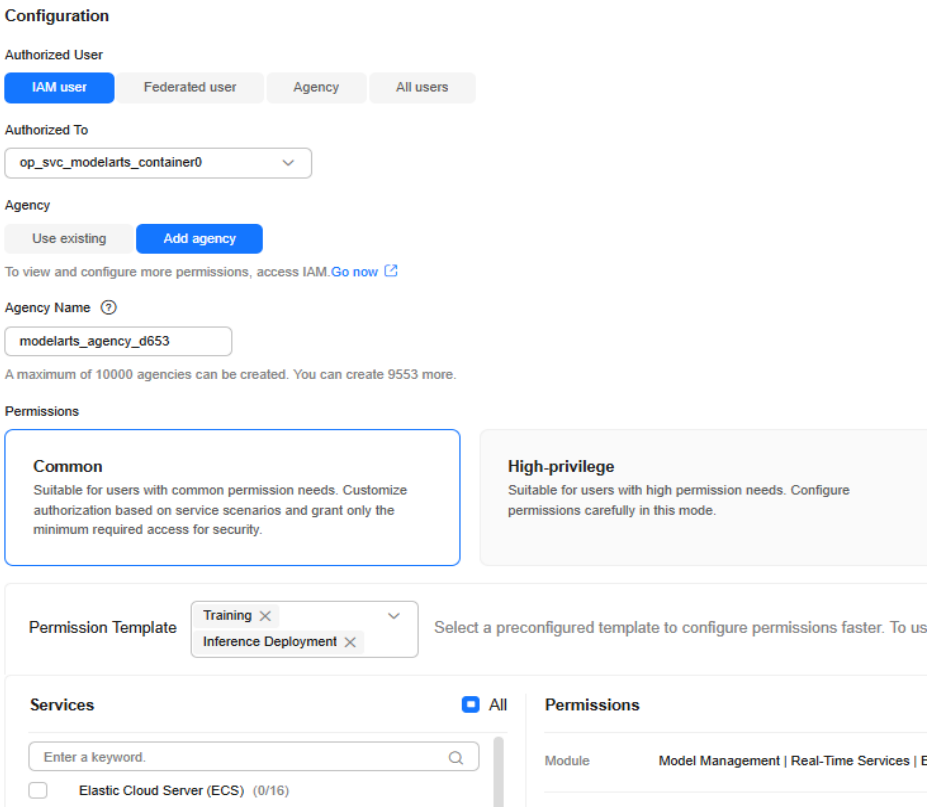
 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Configure ModelArts agency permissions to allow ModelArts to access dependent services such as OBS.

1. Log in to the **ModelArts console** using your Huawei Cloud account. In the navigation pane on the left, choose **Permission Management**. On the **Permission Management** page, click **Add Authorization**.
2. In the displayed dialog box, configure the following parameters:
   - **Authorized User**: **IAM user**.
   - **Agency**: Select **Add agency**.
   - **Permissions**: **Common**.
   - **Permission Template**: Select **Training** and **Inference Deployment**.
   - **Services**: The required minimum permissions are automatically selected based on the items selected in **Permission Template**.

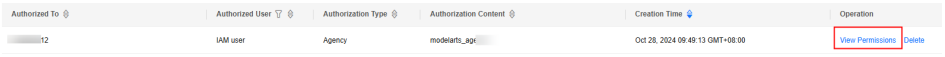
   Select "I have read and agree to the ModelArts Service Statement" and click **Create**.

   Figure 7-1 Configuring an agency



3. After the configuration, view the agency configurations of your account on the **Permission Management** page.

   Figure 7-2 Viewing agency configurations



 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Step 3 Configure SWR organization permissions.

Grant permissions to IAM users in an organization so that they can read, edit, and manage all images in the organization.

Only accounts and IAM users who have the **Manage** permission can add permissions for other users.

1. Log in to the **SWR console**.
2. In the navigation pane on the left, choose **Organization Management**. On the displayed page, click the target organization in the list.
3. In the **Users** tab, click **Add Permission**. In the displayed dialog box, enter an IAM username, select permissions for the user and click **OK**.

For details about SWR authorization management, see **Granting Permissions of an Organization**.

NOTE

You need to grant the SWR organization permission to the IAM user if they do not have the SWR Admin permission.

Step 4 Verify user permissions.

You may need to wait for 30 minutes before the permission settings are applied. Then, verify the configuration.

1. Log in to the **ModelArts console** as an IAM in **UserGroup-2**. On the login page, ensure that **IAM User Login** is selected.

   Change the password as prompted upon the first login.
2. Verify ModelArts permissions.
   a. In the upper left corner of the service list, choose ModelArts. The **ModelArts console** is displayed.
   b. On the ModelArts management console, check whether you can create notebook instances and training jobs, and register images.
3. Verify SFS permissions.
   a. In the upper left corner of the service list, choose SFS. The **SFS console** is displayed.
   b. Click **Create File System** in the upper right corner. If this operation is successful, you have obtained OBS operation permissions.
4. Verify ECS permissions.
   a. In the upper left corner of the service list, choose ECS. The **ECS console** is displayed.
   b. Click **Buy ECS** in the upper right corner. If this operation is successful, you have obtained ECS operation permissions.
5. Verify VPC permissions.
   a. In the upper left corner of the service list, choose VPC. The **VPC console** is displayed.
   b. Click **Create VPC** in the upper right corner. If this operation is successful, you have obtained VPC operation permissions.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

6. Verify DEW permissions.
   a. In the upper left corner of the service list, choose DEW. The **DEW console** is displayed.
   b. Choose **Key Pair Service** > **Private Key Pairs** and click **Create Key Pair**. If this operation is successful, you have obtained DEW operation permissions.
7. Verify OBS permissions.
   a. In the upper left corner of the service list, choose OBS. The **OBS console** is displayed.
   b. Click **Create Bucket** in the upper right corner. If this operation is successful, you have obtained OBS operation permissions.
8. Verify SWR permissions.
   a. In the upper left corner of the service list, choose SWR. The **SWR console** is displayed.
   b. If an SWR page can be properly displayed, you have obtained SWR operation permissions.
   c. Click **upload an image** in the upper right corner. If authorized organizations are displayed, you have obtained the SWR organization permission.

----End

Creating a Dedicated Resource Pool

ModelArts provides dedicated compute resources, which can be used for notebook instances, training jobs, and model deployment. The resources provided in a dedicated resource pool are exclusive, featuring higher resource efficiency than a public resource pool. To use a dedicated resource pool, create one. For details, see **Creating a Dedicated Resource Pool**.

- **Network**: Select the network that has been connected to the VPC. To create a network and interconnect with VPC, see **Configuring the Dedicated Resource Pool to Access the Internet**.
- **Specifications Type** and **Nodes**: Configure the values based on your needs.

Mounting an SFS Turbo File System to an ECS

After you mount an SFS Turbo file system to an ECS, you can upload the training data to SFS Turbo through the ECS. To do so, perform the following operations:

1. Check the cloud service environment.
   - The ECS and the shared SFS disk belong to the same VPC or interconnected VPCs.
   - The base image of the ECS server is Ubuntu 18.04.
   - The ECS and SFS Turbo are in the same subnet.
2. Set the Huawei Cloud image source in the ECS.
   ```
   sudo sed -i "s@http://.*archive.ubuntu.com@http://repo.huaweicloud.com@g" /etc/apt/sources.list
   sudo sed -i "s@http://.*security.ubuntu.com@http://repo.huaweicloud.com@g" /etc/apt/sources.list
   ```
3. Install the NFS client and mount the target disk.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
sudo apt-get update
sudo apt-get install nfs-common
```

4. Obtain the command for mounting the SFS Turbo file system.
   a. Log in to the **SFS console**.
   b. In the navigation pane on the left, choose **SFS Turbo** > **File Systems**. Then, click the file system name to view its details.
   c. In the **Basic Info** tab, obtain and copy the Linux mounting command.
5. Mount the NFS storage to the ECS server.

   Ensure that the corresponding directory exists and run the following commands:

```
mkdir -p /mnt/sfs_turbo
mount -t nfs -o vers=3,nolock 192.168.0.169:/ /mnt/sfs_turbo
```

Granting the Read Permission to ModelArts Users on the ECS

When a custom image is used for training on ModelArts, the default user is **ma-user** and the default user group is **ma-group**. If a file in the ECS is called during training, grant the read permission to **ma-user**. Otherwise, error "Permission denied" will be displayed.

1. Create **ma-user** and **ma-group** in the ECS.

```
default_user=$(getent passwd 1000 | awk -F ':' '{print $1}') || echo "uid: 1000 does not exist" && \
default_group=$(getent group 100 | awk -F ':' '{print $1}') || echo "gid: 100 does not exist" && \
if [ ! -z ${default_group} ] && [ ${default_group} != "ma-group" ]; then \
    groupdel -f ${default_group}; \
    groupadd -g 100 ma-group; \
fi && \
if [ -z ${default_group} ]; then \
    groupadd -g 100 ma-group; \
fi && \
if [ ! -z ${default_user} ] && [ ${default_user} != "ma-user" ]; then \
    userdel -r ${default_user}; \
    useradd -d /home/ma-user -m -u 1000 -g 100 -s /bin/bash ma-user; \
    chmod -R 750 /home/ma-user; \
fi && \
if [ -z ${default_user} ]; then \
    useradd -d /home/ma-user -m -u 1000 -g 100 -s /bin/bash ma-user; \
    chmod -R 750 /home/ma-user; \
fi && \
# set bash as default
rm /bin/sh && ln -s /bin/bash /bin/sh
```

2. Check the created user information.

```
id ma-user
```

   If the following information is displayed, the creation is successful:

```
uid=1000(ma-user) gid=100(ma-group) groups=100(ma-group)
```

Installing and Configuring the OBS CLI

obsutil is a command line tool for accessing and managing OBS. You can use this tool to perform common configuration and management operations on OBS, such as creating buckets, as well as uploading, downloading, and deleting files/folders.

For details about how to install and configure obsutil, see **obsutil Quick Start**.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

NOTICE

Replace the AK/SK and endpoint in the commands with the actual values.

(Optional) Configuring Workspaces

ModelArts allows you to set fine-grained permissions for IAM users and resource isolation between different workspaces. ModelArts workspaces support project resource isolation and settlement of different projects.

If you have enabled enterprise project management, you can bind an enterprise project ID when creating a workspace, add a user group to the enterprise project, and set fine-grained permissions for users in the group.

If you have not enabled enterprise project management, you can create your own workspace on ModelArts, but enterprise project features will remain unavailable.

7.1.3 Running a Single-Node Single-PU Training Job on ModelArts

Process

1. Preparations
   a. **Purchasing Service Resources** (OBS and SWR)
   b. **Assigning Permissions**
   c. **Creating a Dedicated Resource Pool** (The VPC does not need to be connected.)
   d. **Installing and Configuring the OBS CLI**
   e. **(Optional) Configuring Workspaces**
2. Model training
   a. **Building and Debugging an Image Locally**
   b. **Uploading an Image**
   c. **Uploading Data and Algorithms to OBS**
   d. **Debugging Code with Notebook**
   e. **Creating a Single-Node Single-PU Training Job**
   f. **Monitoring Resources**

Building and Debugging an Image Locally

In this section, the Conda environment is packaged to set up the runtime. You can also install Conda dependencies manually using **pip install** or **conda install**.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

NOTE

- The container image should be smaller than 15 GB. For details, see **Constraints on Custom Images of the Training Framework**.
- Build an image through the official open-source website, for example, PyTorch.
- Containers should be built by layer. Each layer must not have more than 1 GB of capacity or 100,000 files. You need to start with the layers that change less frequently. For example, build the OS, CUDA driver, Python, PyTorch, and other dependency packages in sequence.
- If the training data and code change frequently, do not store them in the container image in case you need to build container images frequently.
- The containers can meet the isolation requirements. Do not create conda environments in a container.

1. Export the conda environment.
   a. Start the offline container image:
      ```
      # run on terminal
      docker run -ti ${your_image:tag}
      ```
   b. Obtain **pytorch.tar.gz**:
      ```
      # run on container

      # Create a conda environment named pytorch based on the target base environment.
      conda create --name pytorch --clone base

      pip install conda-pack

      # Pack pytorch env to generate pytorch.tar.gz.
      conda pack -n pytorch -o pytorch.tar.gz
      ```
   c. Upload the package to a local path.
      ```
      # run on terminal
      docker cp ${your_container_id}:/xxx/xxx/pytorch.tar.gz .
      ```
   d. Upload **pytorch.tar.gz** to OBS and **set it to public read**. Obtain, decompress, and clear **pytorch.tar.gz** using the **wget** commands during creation.
2. Create an image.

   Choose either the official Ubuntu 18.04 image or the image with the CUDA driver from NVIDIA as the base image. Obtain the images on the Docker Hub official website.

   To create an image, do as follows: Install the required apt package and driver, configure the **ma-user** user, import the conda environment, and configure the notebook dependency.

   NOTE

   - Creating images with a Dockerfile is recommended. This ensures Dockerfile traceability and archiving, as well as image content without redundancy or residue.
   - To reduce the final image size, delete intermediate files such as TAR packages when building each layer. For details about how to clear the cache, see **conda clean**.
3. Refer to the following example.

   Dockerfile example:
   ```
   FROM nvidia/cuda:11.3.1-cudnn8-devel-ubuntu18.04

   USER root

   # section1: add user ma-user whose uid is 1000 and user group ma-group whose gid is 100. If there already exists 1000:100 but not ma-user:ma-group, below code will remove it
   ```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
RUN default_user=$(getent passwd 1000 | awk -F ':' '{print $1}') || echo "uid: 1000 does not exist" && \
    default_group=$(getent group 100 | awk -F ':' '{print $1}') || echo "gid: 100 does not exist" && \
    if [ ! -z ${default_group} ] && [ ${default_group} != "ma-group" ]; then \
        groupdel -f ${default_group}; \
        groupadd -g 100 ma-group; \
    fi && \
    if [ -z ${default_group} ]; then \
        groupadd -g 100 ma-group; \
    fi && \
    if [ ! -z ${default_user} ] && [ ${default_user} != "ma-user" ]; then \
        userdel -r ${default_user}; \
        useradd -d /home/ma-user -m -u 1000 -g 100 -s /bin/bash ma-user; \
        chmod -R 750 /home/ma-user; \
    fi && \
    if [ -z ${default_user} ]; then \
        useradd -d /home/ma-user -m -u 1000 -g 100 -s /bin/bash ma-user; \
        chmod -R 750 /home/ma-user; \
    fi && \
    # set bash as default
    rm /bin/sh && ln -s /bin/bash /bin/sh

# section2: config apt source and install tools needed.
RUN sed -i "s@http://.*archive.ubuntu.com@http://repo.huaweicloud.com@g" /etc/apt/sources.list && \
    sed -i "s@http://.*security.ubuntu.com@http://repo.huaweicloud.com@g" /etc/apt/sources.list && \
    apt-get update && \
    apt-get install -y ca-certificates curl ffmpeg git libgl1-mesa-glx libglib2.0-0 libibverbs-dev libjpeg-
dev libpng-dev libsm6 libxext6 libxrender-dev ninja-build screen sudo vim wget zip && \
    apt-get clean  && \
    rm -rf /var/lib/apt/lists/*

USER ma-user

# section3: install miniconda and rebuild conda env
RUN mkdir -p /home/ma-user/work/ && cd /home/ma-user/work/ && \
    wget https://repo.anaconda.com/miniconda/Miniconda3-py37_4.12.0-Linux-x86_64.sh && \
    chmod 777 Miniconda3-py37_4.12.0-Linux-x86_64.sh && \
    bash Miniconda3-py37_4.12.0-Linux-x86_64.sh -bfp /home/ma-user/anaconda3 && \
    wget https://${bucketname}.obs.cn-north-4.myhuaweicloud.com/${folder_name}/pytorch.tar.gz && \
    mkdir -p /home/ma-user/anaconda3/envs/pytorch && \
    tar -xzf pytorch.tar.gz -C /home/ma-user/anaconda3/envs/pytorch && \
    source /home/ma-user/anaconda3/envs/pytorch/bin/activate && conda-unpack && \
    /home/ma-user/anaconda3/bin/conda init bash && \
    rm -rf /home/ma-user/work/*

ENV PATH=/home/ma-user/anaconda3/envs/pytorch/bin:$PATH

# section4: settings of Jupyter Notebook for pytorch env
RUN source /home/ma-user/anaconda3/envs/pytorch/bin/activate && \
    pip install ipykernel==6.7.0 --trusted-host https://repo.huaweicloud.com -i https://
repo.huaweicloud.com/repository/pypi/simple && \
    ipython kernel install --user --env PATH /home/ma-user/anaconda3/envs/pytorch/bin:$PATH --
name=pytorch && \
    rm -rf /home/ma-user/.local/share/jupyter/kernels/pytorch/logo-* && \
    rm -rf ~/.cache/pip/* && \
    echo 'export PATH=$PATH:/home/ma-user/.local/bin' >> /home/ma-user/.bashrc && \
    echo 'export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/nvidia/lib64' >> /home/ma-
user/.bashrc && \
    echo 'conda activate pytorch' >> /home/ma-user/.bashrc

ENV DEFAULT_CONDA_ENV_NAME=pytorch
```

NOTE

Replace **https://${bucket_name}.obs.cn-north-4.myhuaweicloud.com/${folder_name}/pytorch.tar.gz** in the Dockerfile with the OBS path of **pytorch.tar.gz** in 1 (the file must be set to public read).

Go to the Dockerfile directory and run the following commands to create an image:

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
# Use the cd command to navigate to the directory that contains the Dockerfile and run the build command.
# docker build -t ${image_name}:${image_version} .
# Example:
docker build -t pytorch-1.13-cuda11.3-cudnn8-ubuntu18.04:v1 .
```

4. Debug an image.

   NOTE

   It is recommended that you use a Dockerfile to include the changes made during debugging in the official container building process and test it again.

   a. Ensure that the corresponding script, code, and process are running properly on the Linux server.

      If the running fails, commission the container and then create a container image.

   b. Ensure that the image file is located correctly and that you have the required file permission.

      Before training, verify that the custom dependency package is normal, that the required packages are in the pip list, and that the Python used by the container is the correct one. (If you have multiple Pythons installed in the container image, you need to set the Python path environment variable.)

   c. Test the training boot script.

      i. Data copy and verification

         Generally, the image does not contain training data and code. You need to copy the required files to the image after starting the image. To avoid running out of disk space, store data, code, and intermediate data in the **/cache** directory. It is recommended that the Linux server have sufficient memory (more than 8 GB) and hard disk (more than 100 GB).

         The following command enables file interaction between Docker and Linux:

         ```
         docker cp data/ 39c9ceedb1f6:/cache/
         ```

         Once you have prepared the data, run the training script and verify that the training starts correctly. Generally, the boot script is as follows:

         ```
         cd /cache/code/
         python start_train.py
         ```

         To troubleshoot the training process, you can access the logs and errors in the container instance and adjust the code and environment variables accordingly.

      ii. Preset script testing

         The **run.sh** script is typically used to copy data and code from OBS to containers, and to copy output results from containers to OBS.

         You can edit and iterate the script in the container instance if the preset script does not produce the desired result.

      iii. Dedicated pool scenario

         Mounting SFS in dedicated pools allows you to import code and data without worrying about OBS operations.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

You can either mount the SFS directory to the **/mnt/sfs_turbo** directory of the debugging node, or sync the directory content with the SFS disk.

To start a container instance during debugging, use the **-v** parameter to mount a directory from the host machine to the container environment.

```
docker run -ti -d -v /mnt/sfs_turbo:/sfs my_deeplearning_image:v1
```

The command above mounts the **/mnt/sfs_turbo** directory of the host machine to the **/sfs** directory of the container. Any changes in the corresponding directories of the host machine and container are synchronized in real time.

d. To locate faults, check the logs for the training image, and check the API response for the inference image.

Run the following command to view all stdout logs output by the container:

```
docker logs -f 39c9ceedb1f6
```

Some logs are stored in the container when creating an inference image. You need to access the container to view these logs. Note: You need to check whether the logs contain errors (including when the container is started and when the API is executed).

e. To modify the user groups of some files that are inconsistent, run the following command as the **root** user on the host machine.

```
docker exec -u root:root 39c9ceedb1f6 bash -c "chown -R ma-user:ma-user /cache"
```

f. To fix an error during debugging, edit it in the container instance. You can run the **commit** command to save the changes.

Uploading an Image

Uploading an image through the client is to run Docker commands on the machine where container engine client is installed to push the image to an image repository of SWR.

If your container engine client is an ECS or CCE node, you can push an image over two types of networks.

- If your client and the image repository are in the same region, you can push an image over private networks.
- If your client and the image repository are in different regions, you can push an image over public networks and the client needs to be bound to an EIP.

NOTE

- Each image layer uploaded through the client cannot be larger than 10 GB.
- Your container engine client version must be 1.11.2 or later.

1. Access SWR.

   a. Log in to the SWR console.

   b. Click **Create Organization** in the upper right corner and enter an organization name to create an organization. Customize the organization name. Replace the organization name **deep-learning** in subsequent commands with the actual organization name.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

c. In the navigation pane on the left, choose **Dashboard** and click **Generate Login Command** in the upper right corner. On the displayed page, click to copy the login command.

NOTE

- The validity period of the generated login command is 24 hours. To obtain a long-term valid login command, see **Obtaining a Long-Term Login or Image Push/Pull Command**. After you obtain a long-term valid login command, your temporary login commands will still be valid as long as they are in their validity periods.
- The domain name at the end of the login command is the image repository address. Record the address for later use.

d. Run the login command on the machine where the container engine is installed.

The message **Login Succeeded** will be displayed upon a successful login.

2. Run the following command on the device where the container engine is installed to label the image:

docker tag *[image_name_1:tag_1] [image_repository_address]/[organization_name]/[image_name_2:tag_2]*

- *[image_name_1:tag_1]*: Replace it with the actual name and tag of the image to be uploaded.
- [image_repository_address]: You can query the address on the SWR console, that is, the domain name at the end of the login command in **1.c**.
- *[organization_name]*: Replace it with the name of the organization created.
- *[image_name_2:tag_2]*: Replace it with the desired image name and tag.

Example:

```
docker tag ${image_name}:${image_version} swr.cn-north-4.myhuaweicloud.com/${organization_name}/${image_name}:${image_version}
```

3. Upload the image to the image repository.

docker push [*image_repository_address*]/[*organization_name*]/[*image_name_2:tag 2*]

Example:

```
docker push swr.cn-north-4.myhuaweicloud.com/${organization_name}/${image_name}:${image_version}
```

To view the pushed image, go to the SWR console and refresh the **My Images** page.

Uploading Data and Algorithms to OBS

- A parallel file system has been created on OBS. For details, see **Creating a Parallel File System**.
- obsutil has been installed and configured. For details, see **Installing and Configuring the OBS CLI**.

Step 1 Prepare data.

1. Prepare dataset samples, decompress them, and use obsutil to upload the dataset to an OBS bucket.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
./obsutil cp ./dog_cat_1w obs://${your_obs_buck}/demo/ -f -r
```

NOTE

OBS supports multiple file upload modes. If the number of files is less than 100, you can upload them on the OBS console. If the number of files is greater than 100, you are advised to use **OBS Browser+** (Windows) and **obsutil** (Linux). The preceding example shows how to use obsutil.

Step 2 Prepare an algorithm.

Upload the **main.py** file to the **demo** folder in the OBS bucket. The **main.py** file contains the following content:

```
import argparse
import os
import random
import shutil
import time
import warnings
from enum import Enum
import torch
import torch.nn as nn
import torch.nn.parallel
import torch.backends.cudnn as cudnn
import torch.distributed as dist
import torch.optim
from torch.optim.lr_scheduler import StepLR
import torch.multiprocessing as mp
import torch.utils.data
import torch.utils.data.distributed
import torchvision.transforms as transforms
import torchvision.datasets as datasets
import torchvision.models as models
model_names = sorted(name for name in models.__dict__
                     if name.islower() and not name.startswith("__")
                     and callable(models.__dict__[name]))
parser = argparse.ArgumentParser(description='PyTorch ImageNet Training')
parser.add_argument('data', metavar='DIR', default='imagenet',
                    help='path to dataset (default: imagenet)')
parser.add_argument('-a', '--arch', metavar='ARCH', default='resnet18',
                    choices=model_names,
                    help='model architecture: ' +
                        ' | '.join(model_names) +
                        ' (default: resnet18)')
parser.add_argument('-j', '--workers', default=4, type=int, metavar='N',
                    help='number of data loading workers (default: 4)')
parser.add_argument('--epochs', default=90, type=int, metavar='N',
                    help='number of total epochs to run')
parser.add_argument('--start-epoch', default=0, type=int, metavar='N',
                    help='manual epoch number (useful on restarts)')
parser.add_argument('-b', '--batch-size', default=256, type=int,
                    metavar='N',
                    help='mini-batch size (default: 256), this is the total '
                        'batch size of all GPUs on the current node when '
                        'using Data Parallel or Distributed Data Parallel')
parser.add_argument('--lr', '--learning-rate', default=0.1, type=float,
                    metavar='LR', help='initial learning rate', dest='lr')
parser.add_argument('--momentum', default=0.9, type=float, metavar='M',
                    help='momentum')
parser.add_argument('--wd', '--weight-decay', default=1e-4, type=float,
                    metavar='W', help='weight decay (default: 1e-4)',
                    dest='weight_decay')
parser.add_argument('-p', '--print-freq', default=10, type=int,
                    metavar='N', help='print frequency (default: 10)')
parser.add_argument('--resume', default='', type=str, metavar='PATH',
                    help='path to latest checkpoint (default: none)')
parser.add_argument('-e', '--evaluate', dest='evaluate', action='store_true',
                    help='evaluate model on validation set')
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
parser.add_argument('--pretrained', dest='pretrained', action='store_true',
                    help='use pre-trained model')
parser.add_argument('--world-size', default=-1, type=int,
                    help='number of nodes for distributed training')
parser.add_argument('--rank', default=-1, type=int,
                    help='node rank for distributed training')
parser.add_argument('--dist-url', default='tcp://224.66.41.62:23456', type=str,
                    help='url used to set up distributed training')
parser.add_argument('--dist-backend', default='nccl', type=str,
                    help='distributed backend')
parser.add_argument('--seed', default=None, type=int,
                    help='seed for initializing training. ')
parser.add_argument('--gpu', default=None, type=int,
                    help='GPU id to use.')
parser.add_argument('--multiprocessing-distributed', action='store_true',
                    help='Use multi-processing distributed training to launch '
                         'N processes per node, which has N GPUs. This is the '
                         'fastest way to use PyTorch for either single node or '
                         'multi node data parallel training')
best_acc1 = 0


def main():
    args = parser.parse_args()
    if args.seed is not None:
        random.seed(args.seed)
        torch.manual_seed(args.seed)
        cudnn.deterministic = True
        warnings.warn('You have chosen to seed training. '
                      'This will turn on the CUDNN deterministic setting, '
                      'which can slow down your training considerably! '
                      'You may see unexpected behavior when restarting '
                      'from checkpoints.')
    if args.gpu is not None:
        warnings.warn('You have chosen a specific GPU. This will completely '
                      'disable data parallelism.')
    if args.dist_url == "env://" and args.world_size == -1:
        args.world_size = int(os.environ["WORLD_SIZE"])
    args.distributed = args.world_size > 1 or args.multiprocessing_distributed
    ngpus_per_node = torch.cuda.device_count()
    if args.multiprocessing_distributed:
        # Since we have ngpus_per_node processes per node, the total world_size
        # needs to be adjusted accordingly
        args.world_size = ngpus_per_node * args.world_size
        # Use torch.multiprocessing.spawn to launch distributed processes: the
        # main_worker process function
        mp.spawn(main_worker, nprocs=ngpus_per_node, args=(ngpus_per_node, args))
    else:
        # Simply call main_worker function
        main_worker(args.gpu, ngpus_per_node, args)
def main_worker(gpu, ngpus_per_node, args):
    global best_acc1
    args.gpu = gpu
    if args.gpu is not None:
        print("Use GPU: {} for training".format(args.gpu))
    if args.distributed:
        if args.dist_url == "env://" and args.rank == -1:
            args.rank = int(os.environ["RANK"])
        if args.multiprocessing_distributed:
            # For multiprocessing distributed training, rank needs to be the
            # global rank among all the processes
            args.rank = args.rank * ngpus_per_node + gpu
        dist.init_process_group(backend=args.dist_backend, init_method=args.dist_url,
                                world_size=args.world_size, rank=args.rank)
    # create model
    if args.pretrained:
        print("=> using pre-trained model '{}'".format(args.arch))
        model = models.__dict__[args.arch](pretrained=True)
    else:
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
    print("=> creating model '{}'".format(args.arch))
    model = models.__dict__[args.arch]()
if not torch.cuda.is_available():
    print('using CPU, this will be slow')
elif args.distributed:
    # For multiprocessing distributed, DistributedDataParallel constructor
    # should always set the single device scope, otherwise,
    # DistributedDataParallel will use all available devices.
    if args.gpu is not None:
        torch.cuda.set_device(args.gpu)
        model.cuda(args.gpu)
        # When using a single GPU per process and per
        # DistributedDataParallel, we need to divide the batch size
        # ourselves based on the total number of GPUs of the current node.
        args.batch_size = int(args.batch_size / ngpus_per_node)
        args.workers = int((args.workers + ngpus_per_node - 1) / ngpus_per_node)
        model = torch.nn.parallel.DistributedDataParallel(model, device_ids=[args.gpu])
    else:
        model.cuda()
        # DistributedDataParallel will divide and allocate batch_size to all
        # available GPUs if device_ids are not set
        model = torch.nn.parallel.DistributedDataParallel(model)
elif args.gpu is not None:
    torch.cuda.set_device(args.gpu)
    model = model.cuda(args.gpu)
else:
    # DataParallel will divide and allocate batch_size to all available GPUs
    if args.arch.startswith('alexnet') or args.arch.startswith('vgg'):
        model.features = torch.nn.DataParallel(model.features)
        model.cuda()
    else:
        model = torch.nn.DataParallel(model).cuda()
# define loss function (criterion), optimizer, and learning rate scheduler
criterion = nn.CrossEntropyLoss().cuda(args.gpu)
optimizer = torch.optim.SGD(model.parameters(), args.lr,
                            momentum=args.momentum,
                            weight_decay=args.weight_decay)
"""Sets the learning rate to the initial LR decayed by 10 every 30 epochs"""
scheduler = StepLR(optimizer, step_size=30, gamma=0.1)
# optionally resume from a checkpoint
if args.resume:
    if os.path.isfile(args.resume):
        print("=> loading checkpoint '{}'".format(args.resume))
        if args.gpu is None:
            checkpoint = torch.load(args.resume)
        else:
            # Map model to be loaded to specified single gpu.
            loc = 'cuda:{}'.format(args.gpu)
            checkpoint = torch.load(args.resume, map_location=loc)
        args.start_epoch = checkpoint['epoch']
        best_acc1 = checkpoint['best_acc1']
        if args.gpu is not None:
            # best_acc1 may be from a checkpoint from a different GPU
            best_acc1 = best_acc1.to(args.gpu)
        model.load_state_dict(checkpoint['state_dict'])
        optimizer.load_state_dict(checkpoint['optimizer'])
        scheduler.load_state_dict(checkpoint['scheduler'])
        print("=> loaded checkpoint '{}' (epoch {})"
              .format(args.resume, checkpoint['epoch']))
    else:
        print("=> no checkpoint found at '{}'".format(args.resume))
cudnn.benchmark = True

# Data loading code
traindir = os.path.join(args.data, 'train')
valdir = os.path.join(args.data, 'val')
normalize = transforms.Normalize(mean=[0.485, 0.456, 0.406],
                                 std=[0.229, 0.224, 0.225])
train_dataset = datasets.ImageFolder(
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
        traindir,
        transforms.Compose([
            transforms.RandomResizedCrop(224),
            transforms.RandomHorizontalFlip(),
            transforms.ToTensor(),
            normalize,
        ]))
    if args.distributed:
        train_sampler = torch.utils.data.distributed.DistributedSampler(train_dataset)
    else:
        train_sampler = None

    train_loader = torch.utils.data.DataLoader(
        train_dataset, batch_size=args.batch_size, shuffle=(train_sampler is None),
        num_workers=args.workers, pin_memory=True, sampler=train_sampler)
    val_loader = torch.utils.data.DataLoader(
        datasets.ImageFolder(valdir, transforms.Compose([
            transforms.Resize(256),
            transforms.CenterCrop(224),
            transforms.ToTensor(),
            normalize,
        ])),
        batch_size=args.batch_size, shuffle=False,
        num_workers=args.workers, pin_memory=True)
    if args.evaluate:
        validate(val_loader, model, criterion, args)
        return

    for epoch in range(args.start_epoch, args.epochs):
        if args.distributed:
            train_sampler.set_epoch(epoch)
        # train for one epoch
        train(train_loader, model, criterion, optimizer, epoch, args)
        # evaluate on validation set
        acc1 = validate(val_loader, model, criterion, args)
        scheduler.step()
        # remember best acc@1 and save checkpoint
        is_best = acc1 > best_acc1
        best_acc1 = max(acc1, best_acc1)
        if not args.multiprocessing_distributed or (args.multiprocessing_distributed
                                                    and args.rank % ngpus_per_node == 0):
            save_checkpoint({
                'epoch': epoch + 1,
                'arch': args.arch,
                'state_dict': model.state_dict(),
                'best_acc1': best_acc1,
                'optimizer': optimizer.state_dict(),
                'scheduler': scheduler.state_dict()
            }, is_best)
def train(train_loader, model, criterion, optimizer, epoch, args):
    batch_time = AverageMeter('Time', ':6.3f')
    data_time = AverageMeter('Data', ':6.3f')
    losses = AverageMeter('Loss', ':.4e')
    top1 = AverageMeter('Acc@1', ':6.2f')
    top5 = AverageMeter('Acc@5', ':6.2f')
    progress = ProgressMeter(
        len(train_loader),
        [batch_time, data_time, losses, top1, top5],
        prefix="Epoch: [{}]".format(epoch))
    # switch to train mode
    model.train()
    end = time.time()
    for i, (images, target) in enumerate(train_loader):
        # measure data loading time
        data_time.update(time.time() - end)
        if args.gpu is not None:
            images = images.cuda(args.gpu, non_blocking=True)
        if torch.cuda.is_available():
            target = target.cuda(args.gpu, non_blocking=True)
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
        # compute output
        output = model(images)
        loss = criterion(output, target)
        # measure accuracy and record loss
        acc1, acc5 = accuracy(output, target, topk=(1, 5))
        losses.update(loss.item(), images.size(0))
        top1.update(acc1[0], images.size(0))
        top5.update(acc5[0], images.size(0))
        # compute gradient and do SGD step
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        # measure elapsed time
        batch_time.update(time.time() - end)
        end = time.time()
        if i % args.print_freq == 0:
            progress.display(i)
def validate(val_loader, model, criterion, args):
    batch_time = AverageMeter('Time', ':6.3f', Summary.NONE)
    losses = AverageMeter('Loss', ':.4e', Summary.NONE)
    top1 = AverageMeter('Acc@1', ':6.2f', Summary.AVERAGE)
    top5 = AverageMeter('Acc@5', ':6.2f', Summary.AVERAGE)
    progress = ProgressMeter(
        len(val_loader),
        [batch_time, losses, top1, top5],
        prefix='Test: ')
    # switch to evaluate mode
    model.eval()
    with torch.no_grad():
        end = time.time()
        for i, (images, target) in enumerate(val_loader):
            if args.gpu is not None:
                images = images.cuda(args.gpu, non_blocking=True)
            if torch.cuda.is_available():
                target = target.cuda(args.gpu, non_blocking=True)
            # compute output
            output = model(images)
            loss = criterion(output, target)
            # measure accuracy and record loss
            acc1, acc5 = accuracy(output, target, topk=(1, 5))
            losses.update(loss.item(), images.size(0))
            top1.update(acc1[0], images.size(0))
            top5.update(acc5[0], images.size(0))
            # measure elapsed time
            batch_time.update(time.time() - end)
            end = time.time()
            if i % args.print_freq == 0:
                progress.display(i)
        progress.display_summary()
    return top1.avg
def save_checkpoint(state, is_best, filename='checkpoint.pth.tar'):
    torch.save(state, filename)
    if is_best:
        shutil.copyfile(filename, 'model_best.pth.tar')
class Summary(Enum):
    NONE = 0
    AVERAGE = 1
    SUM = 2
    COUNT = 3


class AverageMeter(object):
    """Computes and stores the average and current value"""

    def __init__(self, name, fmt=':f', summary_type=Summary.AVERAGE):
        self.name = name
        self.fmt = fmt
        self.summary_type = summary_type
        self.reset()
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
    def reset(self):
        self.val = 0
        self.avg = 0
        self.sum = 0
        self.count = 0

    def update(self, val, n=1):
        self.val = val
        self.sum += val * n
        self.count += n
        self.avg = self.sum / self.count
    def __str__(self):
        fmtstr = '{name} {val' + self.fmt + '} ({avg' + self.fmt + '})'
        return fmtstr.format(**self.__dict__)
    def summary(self):
        fmtstr = ''
        if self.summary_type is Summary.NONE:
            fmtstr = ''
        elif self.summary_type is Summary.AVERAGE:
            fmtstr = '{name} {avg:.3f}'
        elif self.summary_type is Summary.SUM:
            fmtstr = '{name} {sum:.3f}'
        elif self.summary_type is Summary.COUNT:
            fmtstr = '{name} {count:.3f}'
        else:
            raise ValueError('invalid summary type %r' % self.summary_type)
        return fmtstr.format(**self.__dict__)
class ProgressMeter(object):
    def __init__(self, num_batches, meters, prefix=""):
        self.batch_fmtstr = self._get_batch_fmtstr(num_batches)
        self.meters = meters
        self.prefix = prefix
    def display(self, batch):
        entries = [self.prefix + self.batch_fmtstr.format(batch)]
        entries += [str(meter) for meter in self.meters]
        print('\t'.join(entries))
    def display_summary(self):
        entries = [" *"]
        entries += [meter.summary() for meter in self.meters]
        print(' '.join(entries))
    def _get_batch_fmtstr(self, num_batches):
        num_digits = len(str(num_batches // 1))
        fmt = '{:' + str(num_digits) + 'd}'
        return '[' + fmt + '/' + fmt.format(num_batches) + ']'


def accuracy(output, target, topk=(1,)):
    """Computes the accuracy over the k top predictions for the specified values of k"""
    with torch.no_grad():
        maxk = max(topk)
        batch_size = target.size(0)
        _, pred = output.topk(maxk, 1, True, True)
        pred = pred.t()
        correct = pred.eq(target.view(1, -1).expand_as(pred))
        res = []
        for k in topk:
            correct_k = correct[:k].reshape(-1).float().sum(0, keepdim=True)
            res.append(correct_k.mul_(100.0 / batch_size))
        return res
if __name__ == '__main__':
    main()
```

----End

Debugging Code with Notebook

- Notebook billing is as follows:

  - A running notebook instance will be billed based on used resources. The fees vary depending on your selected resources. For details, see **Product 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.

Follow these steps:

1. Register the image. Log in to the ModelArts console. In the navigation pane on the left, choose **Image Management**. Click **Register**. Set **SWR Source** to the image pushed to SWR. Paste the complete SWR address, or click  to select a private image from SWR for registration, and add **GPU** to **Type**.
2. Log in to the ModelArts console. In the navigation pane on the left, choose **Development Workspace** > **Notebook**.
3. Click **Create Notebook**. On the displayed page, configure the parameters.
   a. Configure basic information of the notebook instance, including its name, description, and auto stop status. For details, see **Table 7-5**.

      Table 7-5 Basic parameters

Parameter	Description
Name	Name of a notebook instance, which can contain 1 to 64 characters, including letters, digits, hyphens (-), and underscores (_).
Description	Brief description of a notebook instance.
Auto Stop	Automatically stops the notebook instance at a specified time. This function is enabled by default. The default value is **1 hour**, indicating that the notebook instance automatically stops after running for 1 hour and its resource billing will stop then. The options are **1 hour**, **2 hours**, **4 hours**, **6 hours**, and **Custom**. You can select **Custom** to specify any integer from 1 to 24 hours.

   b. Select an image and configure resource specifications for the instance.
      - **Image**: In the **Custom Images** tab, select the uploaded custom image.
      - **Resource Type**: Select a created dedicated resource pool based on site requirements.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

- **Instance Specifications**: Select 1-GPU specifications.
- **Storage**: Select **EVS**.

NOTE

To use VS Code to connect to a notebook instance for code debugging, enable **Remote SSH** and select a key pair. For details, see **Connecting to a Notebook Instance Through VS Code**.

4. Click **Next**.
5. Confirm the information and click **Submit**.

   Switch to the notebook instance list. The notebook instance is being created. It will take several minutes before its status changes to **Running**.

   If the created notebook instance fails to be started, refer to the key points for debugging described in **Building and Debugging an Image Locally**.
6. In the notebook instance list, click the instance name. On the instance details page that is displayed, view the instance configuration.
7. Mounting an OBS parallel file system: On the notebook instance details page, click the **Storage** tab, then click **Mount Storage**, and set 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**.
8. View the mounting result on the notebook instance details page.
9. Debug the code.

   Open your notebook instance and access a terminal, and go to the mounting directory set in Step 7.

   ```
   cd /data/demo
   ```

   Command for executing training:

   ```
   /home/ma-user/anaconda3/envs/pytorch/bin/python main.py -a resnet50 -b 128 --epochs 5 dog_cat_1w/
   ```

   The alarm "RequestsDependencyWarning: urllib3 (1.26.8) or chardet (5.0.0)/charset_normalizer (2.0.12) doesn't match a supported version!" does not affect training and can be ignored.

   NOTE

   After debugging in a notebook instance, if the image is modified, you can save the image for subsequent training. For details, see **Saving a Notebook Environment Image**.

Creating a Single-Node Single-PU Training Job

NOTE

For training using a dedicated pool, the mounted directory must be the same as that during debugging.

1. Log in to the ModelArts console and check whether access authorization has been configured for your account. For details, see **Configuring Agency**

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Authorization for ModelArts with One Click. If you have been authorized using access keys, clear the authorization and configure agency authorization.

2. In the navigation pane on the left, choose **Model Training** > **Training Jobs**. The training job list is displayed by default. Click **Create Training Job**.
3. On the **Create Training Job** page, configure parameters and click **Submit**.
   - **Algorithm Type**: **Custom algorithm**
   - **Boot Mode**: **Custom image**
   - **Image**: custom image you have uploaded
   - **Boot Command**:
     ```
     cd ${MA_JOB_DIR}/demo && python main.py -a resnet50 -b 128 --epochs 5 dog_cat_1w/
     ```
     demo (customizable) is the last-level directory of the OBS path.
   - **Resource Pool**: In the **Dedicated Resource Pool** tab, select a GPU dedicated resource pool.
   - **Specifications**: Select 1-GPU specifications.
4. Click **Submit**. On the information confirmation page, check the parameters, and click **OK**.
5. Wait until the training job is created.

   Once you submit the job creation request, the system handles tasks like downloading the container image and code directory, and executing the boot command in the backend. Training jobs take varying amounts of time, from tens of minutes to several hours, depending on the service logic and chosen resources.

Monitoring Resources

You can view the resource usage of compute nodes in the **Resource Usages** window. The data of at most the last three days can be displayed. When the resource usage window is opened, the data is loading and refreshed periodically.

Operation 1: If a training job uses multiple compute nodes, choose a node from the drop-down list box to view its metrics.

Operation 2: Click **cpuUsage**, **memUsage**, **npuMemUsage**, or **npuUtil** to show or hide the usage chart of that metric.

Operation 3: Hover over the graph to view the usage at the specific time.

Table 7-6 Parameters

Parameter	Description
cpuUsage	CPU usage
gpuMemUsage	GPU memory usage
gpuUtil	GPU usage
memUsage	Memory usage
npuMemUsage	NPU memory usage

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Parameter	Description
npuUtil	NPU usage

7.1.4 Running a Single-Node Multi-PU Training Job on ModelArts

Process

1. Preparations
   a. **Purchasing Service Resources** (VPC, SFS, SWR, and ECS)
   b. **Assigning Permissions**
   c. **Creating a Dedicated Resource Pool** (interconnecting the VPC)
   d. **Mounting an SFS Turbo File System to an ECS**
   e. **Granting the Read Permission to ModelArts Users on the ECS**
   f. **Installing and Configuring the OBS CLI**
   g. **(Optional) Configuring Workspaces**
2. Model training
   a. **Building and Debugging an Image Locally**
   b. **Uploading an Image**
   c. **Uploading Data and Algorithms to SFS**
   d. **Debugging Code with Notebook**
   e. **Creating a Single-Node Multi-PU Training Job**

Building and Debugging an Image Locally

In this section, the Conda environment is packaged to set up the runtime. You can also install Conda dependencies manually using **pip install** or **conda install**.

NOTE

- The container image should be smaller than 15 GB. For details, see **Constraints on Custom Images of the Training Framework**.
- Build an image through the official open-source website, for example, PyTorch.
- Containers should be built by layer. Each layer must not have more than 1 GB of capacity or 100,000 files. You need to start with the layers that change less frequently. For example, build the OS, CUDA driver, Python, PyTorch, and other dependency packages in sequence.
- If the training data and code change frequently, do not store them in the container image in case you need to build container images frequently.
- The containers can meet the isolation requirements. Do not create conda environments in a container.

1. Export the conda environment.
   a. Start the offline container image:

```
# run on terminal
docker run -ti ${your_image:tag}
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

b. Obtain **pytorch.tar.gz**:

```
# run on container

# Create a conda environment named pytorch based on the target base environment.
conda create --name pytorch --clone base

pip install conda-pack

# Pack pytorch env to generate pytorch.tar.gz.
conda pack -n pytorch -o pytorch.tar.gz
```

c. Upload the package to a local path.

```
# run on terminal
docker cp ${your_container_id}:/xxx/xxx/pytorch.tar.gz .
```

d. Upload **pytorch.tar.gz** to OBS and **set it to public read**. Obtain, decompress, and clear **pytorch.tar.gz** using the **wget** commands during creation.

2. Create an image.

   Choose either the official Ubuntu 18.04 image or the image with the CUDA driver from NVIDIA as the base image. Obtain the images on the Docker Hub official website.

   To create an image, do as follows: Install the required apt package and driver, configure the **ma-user** user, import the conda environment, and configure the notebook dependency.

   NOTE

   - Creating images with a Dockerfile is recommended. This ensures Dockerfile traceability and archiving, as well as image content without redundancy or residue.
   - To reduce the final image size, delete intermediate files such as TAR packages when building each layer. For details about how to clear the cache, see **conda clean**.

3. Refer to the following example.

   Dockerfile example:

```
FROM nvidia/cuda:11.3.1-cudnn8-devel-ubuntu18.04

USER root

# section1: add user ma-user whose uid is 1000 and user group ma-group whose gid is 100. If there already exists 1000:100 but not ma-user:ma-group, below code will remove it
RUN default_user=$(getent passwd 1000 | awk -F ':' '{print $1}') || echo "uid: 1000 does not exist" && \
    default_group=$(getent group 100 | awk -F ':' '{print $1}') || echo "gid: 100 does not exist" && \
    if [ ! -z ${default_group} ] && [ ${default_group} != "ma-group" ]; then \
        groupdel -f ${default_group}; \
        groupadd -g 100 ma-group; \
    fi && \
    if [ -z ${default_group} ]; then \
        groupadd -g 100 ma-group; \
    fi && \
    if [ ! -z ${default_user} ] && [ ${default_user} != "ma-user" ]; then \
        userdel -r ${default_user}; \
        useradd -d /home/ma-user -m -u 1000 -g 100 -s /bin/bash ma-user; \
        chmod -R 750 /home/ma-user; \
    fi && \
    if [ -z ${default_user} ]; then \
        useradd -d /home/ma-user -m -u 1000 -g 100 -s /bin/bash ma-user; \
        chmod -R 750 /home/ma-user; \
    fi && \
    # set bash as default
    rm /bin/sh && ln -s /bin/bash /bin/sh

# section2: config apt source and install tools needed.
RUN sed -i "s@http://.*archive.ubuntu.com@http://repo.huaweicloud.com@g" /etc/apt/sources.list && \
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
    sed -i "s@http://.*security.ubuntu.com@http://repo.huaweicloud.com@g" /etc/apt/sources.list && \
    apt-get update && \
    apt-get install -y ca-certificates curl ffmpeg git libgl1-mesa-glx libglib2.0-0 libibverbs-dev libjpeg-
dev libpng-dev libsm6 libxext6 libxrender-dev ninja-build screen sudo vim wget zip && \
    apt-get clean  && \
    rm -rf /var/lib/apt/lists/*

USER ma-user

# section3: install miniconda and rebuild conda env
RUN mkdir -p /home/ma-user/work/ && cd /home/ma-user/work/ && \
    wget https://repo.anaconda.com/miniconda/Miniconda3-py37_4.12.0-Linux-x86_64.sh && \
    chmod 777 Miniconda3-py37_4.12.0-Linux-x86_64.sh && \
    bash Miniconda3-py37_4.12.0-Linux-x86_64.sh -bfp /home/ma-user/anaconda3 && \
    wget https://${bucketname}.obs.cn-north-4.myhuaweicloud.com/${folder_name}/pytorch.tar.gz && \
    mkdir -p /home/ma-user/anaconda3/envs/pytorch && \
    tar -xzf pytorch.tar.gz -C /home/ma-user/anaconda3/envs/pytorch && \
    source /home/ma-user/anaconda3/envs/pytorch/bin/activate && conda-unpack && \
    /home/ma-user/anaconda3/bin/conda init bash && \
    rm -rf /home/ma-user/work/*

ENV PATH=/home/ma-user/anaconda3/envs/pytorch/bin:$PATH

# section4: settings of Jupyter Notebook for pytorch env
RUN source /home/ma-user/anaconda3/envs/pytorch/bin/activate && \
    pip install ipykernel==6.7.0 --trusted-host https://repo.huaweicloud.com -i https://
repo.huaweicloud.com/repository/pypi/simple && \
    ipython kernel install --user --env PATH /home/ma-user/anaconda3/envs/pytorch/bin:$PATH --
name=pytorch && \
    rm -rf /home/ma-user/.local/share/jupyter/kernels/pytorch/logo-* && \
    rm -rf ~/.cache/pip/* && \
    echo 'export PATH=$PATH:/home/ma-user/.local/bin' >> /home/ma-user/.bashrc && \
    echo 'export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/nvidia/lib64' >> /home/ma-
user/.bashrc && \
    echo 'conda activate pytorch' >> /home/ma-user/.bashrc

ENV DEFAULT_CONDA_ENV_NAME=pytorch
```

NOTE

Replace **https://${bucket_name}.obs.cn-north-4.myhuaweicloud.com/${folder_name}/pytorch.tar.gz** in the Dockerfile with the OBS path of **pytorch.tar.gz** in 1 (the file must be set to public read).

Go to the Dockerfile directory and run the following commands to create an image:

```
# Use the cd command to navigate to the directory that contains the Dockerfile and run the build
command.
# docker build -t ${image_name}:${image_version} .
# Example:
docker build -t pytorch-1.13-cuda11.3-cudnn8-ubuntu18.04:v1 .
```

4. Debug an image.

NOTE

It is recommended that you use a Dockerfile to include the changes made during debugging in the official container building process and test it again.

a. Ensure that the corresponding script, code, and process are running properly on the Linux server.

   If the running fails, commission the container and then create a container image.

b. Ensure that the image file is located correctly and that you have the required file permission.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Before training, verify that the custom dependency package is normal, that the required packages are in the pip list, and that the Python used by the container is the correct one. (If you have multiple Pythons installed in the container image, you need to set the Python path environment variable.)

c. Test the training boot script.

i. Data copy and verification

Generally, the image does not contain training data and code. You need to copy the required files to the image after starting the image. To avoid running out of disk space, store data, code, and intermediate data in the **/cache** directory. It is recommended that the Linux server have sufficient memory (more than 8 GB) and hard disk (more than 100 GB).

The following command enables file interaction between Docker and Linux:

```
docker cp data/ 39c9ceedb1f6:/cache/
```

Once you have prepared the data, run the training script and verify that the training starts correctly. Generally, the boot script is as follows:

```
cd /cache/code/
python start_train.py
```

To troubleshoot the training process, you can access the logs and errors in the container instance and adjust the code and environment variables accordingly.

ii. Preset script testing

The **run.sh** script is typically used to copy data and code from OBS to containers, and to copy output results from containers to OBS.

You can edit and iterate the script in the container instance if the preset script does not produce the desired result.

iii. Dedicated pool scenario

Mounting SFS in dedicated pools allows you to import code and data without worrying about OBS operations.

You can either mount the SFS directory to the **/mnt/sfs_turbo** directory of the debugging node, or sync the directory content with the SFS disk.

To start a container instance during debugging, use the **-v** parameter to mount a directory from the host machine to the container environment.

```
docker run -ti -d -v /mnt/sfs_turbo:/sfs my_deeplearning_image:v1
```

The command above mounts the **/mnt/sfs_turbo** directory of the host machine to the **/sfs** directory of the container. Any changes in the corresponding directories of the host machine and container are synchronized in real time.

d. To locate faults, check the logs for the training image, and check the API response for the inference image.

Run the following command to view all stdout logs output by the container:

```
docker logs -f 39c9ceedb1f6
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Some logs are stored in the container when creating an inference image. You need to access the container to view these logs. Note: You need to check whether the logs contain errors (including when the container is started and when the API is executed).

e. To modify the user groups of some files that are inconsistent, run the following command as the **root** user on the host machine.

```
docker exec -u root:root 39c9ceedb1f6 bash -c "chown -R ma-user:ma-user /cache"
```

f. To fix an error during debugging, edit it in the container instance. You can run the **commit** command to save the changes.

Uploading an Image

Uploading an image through the client is to run Docker commands on the machine where container engine client is installed to push the image to an image repository of SWR.

If your container engine client is an ECS or CCE node, you can push an image over two types of networks.

- If your client and the image repository are in the same region, you can push an image over private networks.
- If your client and the image repository are in different regions, you can push an image over public networks and the client needs to be bound to an EIP.

NOTE

- Each image layer uploaded through the client cannot be larger than 10 GB.
- Your container engine client version must be 1.11.2 or later.

1. Access SWR.
   a. Log in to the **SWR console**.
   b. Click **Create Organization** in the upper right corner and enter an organization name to create an organization. Customize the organization name. Replace the organization name **deep-learning** in subsequent commands with the actual organization name.
   c. In the navigation pane on the left, choose **Dashboard** and click **Generate Login Command** in the upper right corner. On the displayed page, click  to copy the login command.

      NOTE

      - The validity period of the generated login command is 24 hours. To obtain a long-term valid login command, see **Obtaining a Long-Term Login or Image Push/Pull Command**. After you obtain a long-term valid login command, your temporary login commands will still be valid as long as they are in their validity periods.
      - The domain name at the end of the login command is the image repository address. Record the address for later use.

   d. Run the login command on the machine where the container engine is installed.

      The message **Login Succeeded** will be displayed upon a successful login.
2. Run the following command on the device where the container engine is installed to label the image:

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

docker tag *[image_name_1:tag_1] [image_repository_address]/[organization_name]/[image_name_2:tag_2]*

- *[image_name_1:tag_1]*: Replace it with the actual name and tag of the image to be uploaded.
- *[image_repository_address]*: You can query the address on the SWR console, that is, the domain name at the end of the login command in **1.c**.
- *[organization_name]*: Replace it with the name of the organization created.
- *[image_name_2:tag_2]*: Replace it with the desired image name and tag.

Example:

```
docker tag ${image_name}:${image_version} swr.cn-north-4.myhuaweicloud.com/${organization_name}/${image_name}:${image_version}
```

3. Upload the image to the image repository.

   docker push [*image_repository_address*]/[*organization_name*]/[*image_name_2:tag 2*]

   Example:

   ```
   docker push swr.cn-north-4.myhuaweicloud.com/${organization_name}/${image_name}:${image_version}
   ```

   To view the pushed image, go to the SWR console and refresh the **My Images** page.

Uploading Data and Algorithms to SFS

- SFS has been mounted to the ECS. For details, see **Mounting an SFS Turbo File System to an ECS**.
- Permissions have been configured on the ECS. For details, see **Granting the Read Permission to ModelArts Users on the ECS**.
- obsutil has been installed and configured. For details, see **Installing and Configuring the OBS CLI**.

Step 1 Prepare data.

1. Visit the official COCO dataset website and download the dataset from **https://cocodataset.org/#download**.
2. Download **Train images** (18 GB), **Val images** (1 GB) and **Train/Val annotations** (241 MB) of the 2017 COCO dataset, decompress them, and save them to the **coco** folder.
3. After the download is complete, upload the data to the target directory of SFS. The dataset is very large, so you are advised to use obsutil to upload it to an OBS bucket first, and then move it to SFS.

   a. Run the following commands on the local host and use obsutil to upload the local dataset to an OBS bucket.

   ```
   # Upload local data to OBS.
   # ./obsutil cp ${Path of the local folder where the dataset is stored} ${Path of the OBS folder where the dataset is stored} -f -r
   # Example:
   ./obsutil cp ./coco obs://your_bucket/ -f -r
   ```

   b. Log in to the ECS and use obsutil to migrate the dataset to SFS. The sample code is as follows:

   ```
   # Transfer OBS data to SFS.
   # ./obsutil cp ${Path of the OBS folder where the dataset is located} ${Path of the SFS folder} -f
   ```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
-r
# Example:
./obsutil cp obs://your_bucket/coco/ /mnt/sfs_turbo/ -f -r
```

The directory structure in the **/mnt/sfs_turbo/coco** folder is as follows:

```
coco
|---annotations
|---train2017
|---val2017
```

For more obsutil operations, see **Introduction to obsutil**.

c. Set the file owner to **ma-user**.

```
chown -R ma-user:ma-group coco
```

Step 2 Prepare an algorithm.

1. Download the YOLOX code. Code repository address: **https://github.com/Megvii-BaseDetection/YOLOX.git**.

```
git clone https://github.com/Megvii-BaseDetection/YOLOX.git
cd YOLOX
git checkout 4f8f1d79c8b8e530495b5f183280bab99869e845
```

2. Change the onnx version in the **requirements.txt** file to at least 1.12.0.
3. Change **data_dir = os.path.join(get_yolox_datadir(), "COCO")** in line 59 of **yolox/data/datasets/coco.py** to **data_dir = '/home/ma-user/coco'**.

```
# data_dir = os.path.join(get_yolox_datadir(), "COCO")
data_dir = '/home/ma-user/coco'
```

4. Add the following code before line 13 in **tools/train.py**:

```
# Add the two lines of code to avoid the yolox module not found error during execution.
import sys
sys.path.append(os.getcwd())

# line13
from yolox.core import launch
from yolox.exp import Exp, get_exp
```

5. Change **fast_cocoeval** to **fast_coco_eval_api** in line 122 of **jit_ops.py** in **yolox/layers**.

```
# def __init__(self, name="fast_cocoeval"):
def __init__(self, name="fast_coco_eval_api"):
```

6. Change **from yolox.layers import COCOeval_opt as COCOeval** to **from pycocotools.cocoeval import COCOeval** in line 294 of **coco_evaluator.py** in **yolox\evaluators**.

```
try:
  # from yolox.layers import COCOeval_opt as COCOeval
  from pycocotools.cocoeval import COCOeval
except ImportError:
  from pycocotools.cocoeval import COCOeval

  logger.warning("Use standard COCOeval.")
```

7. Create a **run.sh** script in the **tools** directory and use it as the boot script. The following is a reference for the **run.sh** script:

```
#!/usr/bin/env sh
set -x
set -o pipefail

export NCCL_DEBUG=INFO

DEFAULT_ONE_GPU_BATCH_SIZE=32
BATCH_SIZE=$((${MA_NUM_GPUS:-8} * ${VC_WORKER_NUM:-1} * $
{DEFAULT_ONE_GPU_BATCH_SIZE}))
if [ ${VC_WORKER_HOSTS} ];then
   YOLOX_DIST_URL=tcp://$(echo ${VC_WORKER_HOSTS} | cut -d "," -f 1):6666
   /home/ma-user/anaconda3/envs/pytorch/bin/python -u tools/train.py \
                          -n yolox-s \
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
                        --devices ${MA_NUM_GPUS:-8} \
                        --batch-size ${BATCH_SIZE} \
                        --fp16 \
                        --occupy \
                        --num_machines ${VC_WORKER_NUM:-1} \
                        --machine_rank ${VC_TASK_INDEX:-0} \
                        --dist-url ${YOLOX_DIST_URL}
else
    /home/ma-user/anaconda3/envs/pytorch/bin/python -u tools/train.py \
                        -n yolox-s \
                        --devices ${MA_NUM_GPUS:-8} \
                        --batch-size ${BATCH_SIZE} \
                        --fp16 \
                        --occupy \
                        --num_machines ${VC_WORKER_NUM:-1} \
                        --machine_rank ${VC_TASK_INDEX:-0}
fi
```

NOTE

Some environment variables do not exist in the notebook environment. Default values are required.

8. Save the code to OBS and upload the code to the target directory of SFS through OBS.

   a. Run the following commands on the local host and use obsutil to upload the local dataset to an OBS bucket.

   ```
   # Upload local code to OBS.
   ./obsutil cp ./YOLOX obs://your_bucket/ -f -r
   ```

   b. Log in to the ECS and use obsutil to migrate the dataset to SFS. The sample code is as follows:

   ```
   # Upload code from OBS to SFS.
   ./obsutil cp obs://your_bucket/YOLOX/ /mnt/sfs_turbo/code/ -f -r
   ```

   NOTE

   In this example, obsutils is used to upload files. You can also use SCP to upload files. For details, see **How Can I Use SCP to Transfer Files Between a Local Linux Computer and a Linux ECS?**

9. Set the file owner to **ma-user** in SFS.

   ```
   chown -R ma-user:ma-group YOLOX
   ```

10. Remove **\r** from the shell script:

    ```
    cd YOLOX
    sed -i 's/\r//' run.sh
    ```

    NOTE

    Shell scripts written in Windows have **\r\n** as the line ending, but Linux uses **\n** as the line ending. This means that Linux treats **\r** as part of the script and shows the error message "$'\r': command not found" when running it. To fix this, you need to remove **\r** from the shell script.

----End

Debugging Code with Notebook

- Notebook billing is as follows:
  - A running notebook instance will be billed based on used resources. The fees vary depending on your selected resources. For details, see **Product Pricing Details**. When a notebook instance is not used, stop it.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

  - 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.

Follow these steps:

1. Register the image. Log in to the **ModelArts console**. In the navigation pane on the left, choose **Image Management**. Click **Register**. Set **SWR Source** to the image pushed to SWR. Paste the complete SWR address, or click  to select a private image from SWR for registration, and add **GPU** to **Type**.
2. Log in to the **ModelArts console**. In the navigation pane on the left, choose **Development Workspace** > **Notebook**.
3. Click **Create Notebook**. On the displayed page, configure the parameters.
   a. Configure basic information of the notebook instance, including its name, description, and auto stop status. For details, see **Table 7-7**.

      Table 7-7 Basic parameters

Parameter	Description
Name	Name of a notebook instance, which can contain 1 to 64 characters, including letters, digits, hyphens (-), and underscores (_).
Description	Brief description of a notebook instance.
Auto Stop	Automatically stops the notebook instance at a specified time. This function is enabled by default. The default value is **1 hour**, indicating that the notebook instance automatically stops after running for 1 hour and its resource billing will stop then. The options are **1 hour**, **2 hours**, **4 hours**, **6 hours**, and **Custom**. You can select **Custom** to specify any integer from 1 to 24 hours.

   b. Select an image and configure resource specifications for the instance.
      - **Image**: In the **Custom Images** tab, select the uploaded custom image.
      - **Resource Type**: Select a created dedicated resource pool based on site requirements.
      - **Instance Specifications**: Select 8-GPUs specifications. This matches the default setting of **MA_NUM_GPUS** in the **run.sh** file, which is 8 PUs.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

- **Storage**: Select **SFS**. **Mounted Subdirectory** is optional.

  NOTE

  To use VS Code to connect to a notebook instance for code debugging, enable **Remote SSH** and select a key pair. For details, see **Connecting to a Notebook Instance Through VS Code**.

4. Click **Next**.
5. Confirm the information and click **Submit**.

   Switch to the notebook instance list. The notebook instance is being created. It will take several minutes before its status changes to **Running**.
6. In the notebook instance list, click the instance name. On the instance details page that is displayed, view the instance configuration.
7. Open a terminal in notebook and enter the boot command to debug the code.

```
# Create a dataset soft link.
# ln -s /home/ma-user/work/${Path of the COCO dataset on SFS} /home/ma-user/coco
# Go to the corresponding directory.
# cd /home/ma-user/work/${YOLOX path on SFS}
# Install the environment and run the script.
# /home/ma-user/anaconda3/envs/pytorch/bin/pip install -r requirements.txt && /bin/sh tools/run.sh

# Example:
ln -s /home/ma-user/work/coco /home/ma-user/coco
cd /home/ma-user/work/code/YOLOX/
/home/ma-user/anaconda3/envs/pytorch/bin/pip install -r requirements.txt && /bin/sh tools/run.sh
```

NOTE

After debugging in a notebook instance, if the image is modified, you can save the image for subsequent training. For details, see **Saving a Notebook Environment Image**.

Creating a Single-Node Multi-PU Training Job

1. Log in to the **ModelArts console** and check whether access authorization has been configured for your account. For details, see **Configuring Agency Authorization for ModelArts with One Click**. If you have been authorized using access keys, clear the authorization and configure agency authorization.
2. In the navigation pane on the left, choose **Model Training** > **Training Jobs**. The training job list is displayed by default. Click **Create Training Job**.
3. On the **Create Training Job** page, configure parameters and click **Submit**.
   - **Algorithm Type**: **Custom algorithm**
   - **Boot Mode**: **Custom image**
   - **Image**: custom image you have uploaded
   - **Boot Command**:

     ```
     ln -s /home/ma-user/work/coco /home/ma-user/coco && cd /home/ma-user/work/code/YOLOX/ && /home/ma-user/anaconda3/envs/pytorch/bin/pip install -r requirements.txt && /bin/sh tools/run.sh
     ```

   - **Resource Pool**: In the **Dedicated Resource Pool** tab, select a GPU dedicated resource pool.
   - **Specifications**: Select 8-GPU specifications.
   - **Compute Nodes**: Enter **1**.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

- **SFS Turbo**: Add the mount configuration and select the SFS name. The cloud mount path is **/home/ma-user/work**.

  NOTE

  To ensure that the code path and boot command are the same as those for notebook debugging, set the cloud mount path to **/home/ma-user/work**.

4. Click **Submit**. On the information confirmation page, check the parameters, and click **OK**.
5. Wait until the training job is created.

   Once you submit the job creation request, the system handles tasks like downloading the container image and code directory, and executing the boot command in the backend. Training jobs take varying amounts of time, from tens of minutes to several hours, depending on the service logic and chosen resources.

7.1.5 Running a Multi-Node Multi-PU Training Job on ModelArts

Process

1. Preparations
   a. **Purchasing Service Resources** (VPC, SFS, OBS, SWR, and ECS)
   b. **Assigning Permissions**
   c. **Creating a Dedicated Resource Pool** (connecting to the VPC)
   d. **Mounting an SFS Turbo File System to an ECS**
   e. **Granting the Read Permission to ModelArts Users on the ECS**
   f. **Installing and Configuring the OBS CLI**
   g. **(Optional) Configuring Workspaces**
2. Model training
   a. **Building and Debugging an Image Locally**
   b. **Uploading an Image**
   c. **Uploading data to OBS**
   d. **Uploading the Algorithm to SFS**
   e. **Debugging Code with Notebook**
   f. **Creating a Multi-Node Multi-PU Training Job**

Building and Debugging an Image Locally

In this section, conda env is packaged to build the environment. You can also install conda environment dependencies using **pip install** or **conda install**.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

NOTE

- The container image should be smaller than 15 GB. For details, see **Constraints on Custom Images of the Training Framework**.
- Build an image through the official open-source website, for example, PyTorch.
- Containers should be built by layer. Each layer must not have more than 1 GB of capacity or 100,000 files. You need to start with the layers that change less frequently. For example, build the OS, CUDA driver, Python, PyTorch, and other dependency packages in sequence.
- If the training data and code change frequently, do not store them in the container image in case you need to build container images frequently.
- The containers can meet the isolation requirements. Do not create conda environments in a container.

1. Export the conda environment.
   a. Start the offline container image.
      ```
      # run on terminal
      docker run -ti ${your_image:tag}
      ```
   b. Obtain **pytorch.tar.gz**.
      ```
      # run on container

      # Create a conda environment named pytorch based on the target base environment.
      conda create --name pytorch --clone base

      pip install conda-pack

      # Pack pytorch env to generate pytorch.tar.gz.
      conda pack -n pytorch -o pytorch.tar.gz
      ```
   c. Upload the package to a local path.
      ```
      # run on terminal
      docker cp ${your_container_id}:/xxx/xxx/pytorch.tar.gz .
      ```
   d. Upload **pytorch.tar.gz** to OBS and **set it to public read**. Obtain, decompress, and clear **pytorch.tar.gz** using the **wget** commands during creation.
2. Create an image.

   Choose either the official Ubuntu 18.04 image or the image with the CUDA driver from NVIDIA as the base image. Obtain the images on the Docker Hub official website.

   To create an image, do as follows: Install the required apt package and driver, configure the **ma-user** user, import the conda environment, and configure the notebook dependency.

   NOTE

   - Creating images with a Dockerfile is recommended. This ensures Dockerfile traceability and archiving, as well as image content without redundancy or residue.
   - To reduce the final image size, delete intermediate files such as TAR packages when building each layer. For details about how to clear the cache, see **conda clean**.
3. Refer to the following example.

   Dockerfile example:
   ```
   FROM nvidia/cuda:11.3.1-cudnn8-devel-ubuntu18.04

   USER root

   # section1: add user ma-user whose uid is 1000 and user group ma-group whose gid is 100. If there already exists 1000:100 but not ma-user:ma-group, below code will remove it
   ```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
RUN default_user=$(getent passwd 1000 | awk -F ':' '{print $1}') || echo "uid: 1000 does not exist" && \
    default_group=$(getent group 100 | awk -F ':' '{print $1}') || echo "gid: 100 does not exist" && \
    if [ ! -z ${default_group} ] && [ ${default_group} != "ma-group" ]; then \
        groupdel -f ${default_group}; \
        groupadd -g 100 ma-group; \
    fi && \
    if [ -z ${default_group} ]; then \
        groupadd -g 100 ma-group; \
    fi && \
    if [ ! -z ${default_user} ] && [ ${default_user} != "ma-user" ]; then \
        userdel -r ${default_user}; \
        useradd -d /home/ma-user -m -u 1000 -g 100 -s /bin/bash ma-user; \
        chmod -R 750 /home/ma-user; \
    fi && \
    if [ -z ${default_user} ]; then \
        useradd -d /home/ma-user -m -u 1000 -g 100 -s /bin/bash ma-user; \
        chmod -R 750 /home/ma-user; \
    fi && \
    # set bash as default
    rm /bin/sh && ln -s /bin/bash /bin/sh

# section2: config apt source and install tools needed.
RUN sed -i "s@http://.*archive.ubuntu.com@http://repo.huaweicloud.com@g" /etc/apt/sources.list && \
    sed -i "s@http://.*security.ubuntu.com@http://repo.huaweicloud.com@g" /etc/apt/sources.list && \
    apt-get update && \
    apt-get install -y ca-certificates curl ffmpeg git libgl1-mesa-glx libglib2.0-0 libibverbs-dev libjpeg-
dev libpng-dev libsm6 libxext6 libxrender-dev ninja-build screen sudo vim wget zip && \
    apt-get clean  && \
    rm -rf /var/lib/apt/lists/*

USER ma-user

# section3: install miniconda and rebuild conda env
RUN mkdir -p /home/ma-user/work/ && cd /home/ma-user/work/ && \
    wget https://repo.anaconda.com/miniconda/Miniconda3-py37_4.12.0-Linux-x86_64.sh && \
    chmod 777 Miniconda3-py37_4.12.0-Linux-x86_64.sh && \
    bash Miniconda3-py37_4.12.0-Linux-x86_64.sh -bfp /home/ma-user/anaconda3 && \
    wget https://${bucketname}.obs.cn-north-4.myhuaweicloud.com/${folder_name}/pytorch.tar.gz && \
    mkdir -p /home/ma-user/anaconda3/envs/pytorch && \
    tar -xzf pytorch.tar.gz -C /home/ma-user/anaconda3/envs/pytorch && \
    source /home/ma-user/anaconda3/envs/pytorch/bin/activate && conda-unpack && \
    /home/ma-user/anaconda3/bin/conda init bash && \
    rm -rf /home/ma-user/work/*

ENV PATH=/home/ma-user/anaconda3/envs/pytorch/bin:$PATH

# section4: settings of Jupyter Notebook for pytorch env
RUN source /home/ma-user/anaconda3/envs/pytorch/bin/activate && \
    pip install ipykernel==6.7.0 --trusted-host https://repo.huaweicloud.com -i https://
repo.huaweicloud.com/repository/pypi/simple && \
    ipython kernel install --user --env PATH /home/ma-user/anaconda3/envs/pytorch/bin:$PATH --
name=pytorch && \
    rm -rf /home/ma-user/.local/share/jupyter/kernels/pytorch/logo-* && \
    rm -rf ~/.cache/pip/* && \
    echo 'export PATH=$PATH:/home/ma-user/.local/bin' >> /home/ma-user/.bashrc && \
    echo 'export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/nvidia/lib64' >> /home/ma-
user/.bashrc && \
    echo 'conda activate pytorch' >> /home/ma-user/.bashrc

ENV DEFAULT_CONDA_ENV_NAME=pytorch
```

NOTE

Replace **https://${bucket_name}.obs.cn-north-4.myhuaweicloud.com/${folder_name}/pytorch.tar.gz** in the Dockerfile with the OBS path of **pytorch.tar.gz** in 1 (the file must be set to public read).

Go to the Dockerfile directory and run the following commands to create an image:

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
# Use the cd command to navigate to the directory that contains the Dockerfile and run the build command.
# docker build -t ${image_name}:${image_version} .
# For example:
docker build -t pytorch-1.13-cuda11.3-cudnn8-ubuntu18.04:v1 .
```

4. Debug an image.

   NOTE

   It is recommended that you use a Dockerfile to include the changes made during debugging in the official container building process and test it again.

   a. Ensure that the corresponding script, code, and process are running properly on the Linux server.

      If the running fails, commission the container and then create a container image.

   b. Ensure that the image file is located correctly and that you have the required file permission.

      Before training, verify that the custom dependency package is normal, that the required packages are in the pip list, and that the Python used by the container is the correct one. (If you have multiple Pythons installed in the container image, you need to set the Python path environment variable.)

   c. Test the training boot script.

      i. Data copy and verification

         Generally, the image does not contain training data and code. You need to copy the required files to the image after starting the image. To avoid running out of disk space, store data, code, and intermediate data in the **/cache** directory. It is recommended that the Linux server have sufficient memory (more than 8 GB) and hard disk (more than 100 GB).

         The following command enables file interaction between Docker and Linux:

         ```
         docker cp data/ 39c9ceedb1f6:/cache/
         ```

         Once you have prepared the data, run the training script and verify that the training starts correctly. Generally, the boot script is as follows:

         ```
         cd /cache/code/
         python start_train.py
         ```

         To troubleshoot the training process, you can access the logs and errors in the container instance and adjust the code and environment variables accordingly.

      ii. Preset script testing

         The **run.sh** script is typically used to copy data and code from OBS to containers, and to copy output results from containers to OBS. For details about how to build **run.sh**, see **Running a GPU-based Training Job on ModelArts**.

         You can edit and iterate the script in the container instance if the preset script does not produce the desired result.

      iii. Dedicated pool scenario

         Mounting SFS in dedicated pools allows you to import code and data without worrying about OBS operations.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

You can either mount the SFS directory to the **/mnt/sfs_turbo** directory of the debugging node, or sync the directory content with the SFS disk.

To start a container instance during debugging, use the **-v** parameter to mount a directory from the host machine to the container environment.

```
docker run -ti -d -v /mnt/sfs_turbo:/sfs my_deeplearning_image:v1
```

The command above mounts the **/mnt/sfs_turbo** directory of the host machine to the **/sfs** directory of the container. Any changes in the corresponding directories of the host machine and container are synchronized in real time.

d. To locate faults, check the logs for the training image, and check the API response for the inference image.

Run the following command to view all stdout logs output by the container:

```
docker logs -f 39c9ceedb1f6
```

Some logs are stored in the container when creating an inference image. You need to access the container to view these logs. Note: You need to check whether the logs contain errors (including when the container is started and when the API is executed).

e. To modify the user groups of some files that are inconsistent, run the following command as the **root** user on the host machine.

```
docker exec -u root:root 39c9ceedb1f6 bash -c "chown -R ma-user:ma-user /cache"
```

f. To fix an error during debugging, edit it in the container instance. You can run the **commit** command to save the changes.

Uploading an Image

Uploading an image through the client is to run Docker commands on the machine where container engine client is installed to push the image to an image repository of SWR.

If your container engine client is an ECS or CCE node, you can push an image over two types of networks.

- If your client and the image repository are in the same region, you can push an image over private networks.
- If your client and the image repository are in different regions, you can push an image over public networks and the client needs to be bound to an EIP.

NOTE

- Each image layer uploaded through the client cannot be larger than 10 GB.
- Your container engine client version must be 1.11.2 or later.

1. Access SWR.
   a. Log in to the **SWR console**.
   b. Click **Create Organization** in the upper right corner and enter an organization name to create an organization. Customize the organization name. Replace the organization name **deep-learning** in subsequent commands with the actual organization name.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

c. In the navigation pane on the left, choose **Dashboard** and click **Generate Login Command** in the upper right corner. On the displayed page, click to copy the login command.

NOTE

- The validity period of the generated login command is 24 hours. To obtain a long-term valid login command, see **Obtaining a Login Command with Long-Term Validity**. After you obtain a long-term valid login command, your temporary login commands will still be valid as long as they are in their validity periods.
- The domain name at the end of the login command is the image repository address. Record the address for later use.

d. Run the login command on the machine where the container engine is installed.

The message **Login Succeeded** will be displayed upon a successful login.

2. Run the following command on the device where the container engine is installed to label the image:

docker tag *[Image name 1:tag 1] [Image repository address]/[Organization name]/[Image name 2:tag 2]*

- [Image name 1:tag 1]: Replace it with the actual name and tag of the image to be uploaded.
- [Image repository address]: You can query the address on the SWR console, that is, the domain name at the end of the login command in **1.c**.
- [Organization name]: Replace it with the name of the organization created.
- [Image name 2:tag 2]: Replace it with the desired image name and tag.

Example:

```
docker tag ${image_name}:${image_version} swr.cn-north-4.myhuaweicloud.com/${organization_name}/${image_name}:${image_version}
```

3. Run the following command to push the image to the image repository:

docker push [*Image repository address*]/[*Organization name*]/[*Image name 2:tag 2*]

Example:

```
docker push swr.cn-north-4.myhuaweicloud.com/${organization_name}/${image_name}:${image_version}
```

To view the pushed image, go to the SWR console and refresh the **My Images** page.

Uploading data to OBS

- A common OBS bucket has been created. For details, see **Creating a Bucket**.
- obsutil has been installed. For details, see **Installing and Configuring the OBS CLI**.
- For details about the data transmission principle between OBS and the training container, see **Running a GPU-based Training Job on ModelArts**.

1. Visit the ImageNet official website at **https://image-net.org/** and download the ImageNet-21K dataset.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

2. After converting the format, download the annotation files: **ILSVRC2021winner21k_whole_map_train.txt** and **ILSVRC2021winner21k_whole_map_val.txt**.
3. Upload the preceding three files to the **imagenet21k_whole** folder in the OBS bucket. For details about how to upload files to an OBS bucket, see **Uploading Data and Algorithms to OBS**.

Uploading the Algorithm to SFS

1. Download the Swin-Transformer code.
   ```
   git clone --recursive https://github.com/microsoft/Swin-Transformer.git
   ```
2. Comment out line 27 **t_mul=1** of the **lr_scheduler.py** file.
3. Comment out line 28 **print("ERROR IMG LOADED: ", path)** of the **imagenet22k_dataset.py** file in the **data** folder.
4. Change **prefix = 'ILSVRC2011fall_whole'** to **prefix = 'ILSVRC2021winner21k_whole'** in line 112 of the **build.py** file in the **data** folder.
5. Create the **requirements.txt** file in the **Swin-Transformer** directory to specify the Python dependency library.
   ```
   # The content of the requirements.txt file is as follows:

   timm==0.4.12
   termcolor==1.1.0
   yacs==0.1.8
   ```
6. Prepare the OBS paths required in the **run.sh** file.
   a. Prepare the URL for sharing the ImageNet dataset.

      Select the **imagenet21k_whole** dataset folder to be shared, click the share button, select the validity period of the share URL, enter the access code, for example, **123456**, click **Copy Link**, and record the URL.

   b. Prepare the **obsutil_linux_amd64.tar.gz** share URL.

      Download **obsutil_linux_amd64.tar.gz** by referring to **Downloading and Installing obsutil**, upload it to the OBS bucket, and set it to public read. Click **Object Properties** and copy the URL.

      Sample link:

      ```
      https://${bucketname_name}.obs.cn-north-4.myhuaweicloud.com/${folders_name}/pytorch.tar.gz
      ```
7. In the **Swin-Transformer** directory, create the **run.sh** boot script.

   NOTE

   - Replace **SRC_DATA_PATH=${***URL for sharing the ImageNet dataset in OBS***}** in the script with the share URL of the **imagenet21k_whole** folder in the previous step.
   - Replace **https://${bucket_name}.obs.cn-north-4.myhuaweicloud.com/${folder_name}/obsutil_linux_amd64.tar.gz** in the script with the OBS path of **obsutil_linux_amd64.tar.gz** in the previous step (the file must be set to public read).

   Boot script for single-node single-PU training:

   ```
   # Create a run.sh script in the home directory of the code. The script content is as follows:

   #!/bin/bash

   # Download data from OBS to the local SSD.
   DIS_DATA_PATH=/cache
   SRC_DATA_PATH=${URL for sharing the ImageNet dataset in OBS}
   ```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
OBSUTIL_PATH=https://${bucket_name}.obs.cn-north-4.myhuaweicloud.com/${folder_name}/obsutil_linux_amd64.tar.gz
mkdir -p $DIS_DATA_PATH && cd $DIS_DATA_PATH && wget $OBSUTIL_PATH && tar -xzvf obsutil_linux_amd64.tar.gz && $DIS_DATA_PATH/obsutil_linux_amd64*/obsutil share-cp $SRC_DATA_PATH $DIS_DATA_PATH/ -ac=123456 -r -f -j 256 && cd -
IMAGE_DATA_PATH=$DIS_DATA_PATH/imagenet21k_whole

# Path for storing the model weights and training configurations during model training
OUTPUT_PATH=/cache/output

MASTER_PORT="6061"

/home/ma-user/anaconda3/envs/pytorch/bin/python -m torch.distributed.launch --nproc_per_node=1 --master_addr localhost --master_port=$MASTER_PORT main.py --data-path $IMAGE_DATA_PATH --output $OUTPUT_PATH --cfg ./configs/swin/swin_base_patch4_window7_224_22k.yaml --local_rank 0
```

Boot script for multi-node multi-PU training:

```
# Create a run.sh script.

#!/bin/bash

# Download data from OBS to the local SSD.
DIS_DATA_PATH=/cache
SRC_DATA_PATH=${URL for sharing the ImageNet dataset in OBS}
OBSUTIL_PATH=https://${bucket_name}.obs.cn-north-4.myhuaweicloud.com/${folder_name}/obsutil_linux_amd64.tar.gz
mkdir -p $DIS_DATA_PATH && cd $DIS_DATA_PATH && wget $OBSUTIL_PATH && tar -xzvf obsutil_linux_amd64.tar.gz && $DIS_DATA_PATH/obsutil_linux_amd64*/obsutil share-cp $SRC_DATA_PATH $DIS_DATA_PATH/ -ac=123456 -r -f -j 256 && cd -
IMAGE_DATA_PATH=$DIS_DATA_PATH/imagenet21k_whole

# Path for storing the model weights and training configurations during model training
OUTPUT_PATH=/cache/output

MASTER_ADDR=$(echo ${VC_WORKER_HOSTS} | cut -d "," -f 1)
MASTER_PORT="6060"
NNODES="$VC_WORKER_NUM"
NODE_RANK="$VC_TASK_INDEX"
NGPUS_PER_NODE="$MA_NUM_GPUS"

/home/ma-user/anaconda3/envs/pytorch/bin/python -m torch.distributed.launch --nnodes=$NNODES --node_rank=$NODE_RANK --nproc_per_node=$NGPUS_PER_NODE --master_addr $MASTER_ADDR --master_port=$MASTER_PORT main.py --data-path $IMAGE_DATA_PATH --output=$OUTPUT_PATH --cfg ./configs/swin/swin_base_patch4_window7_224_22k.yaml
```

NOTE

- You are advised to use the boot script for single-node single-PU training. After the script runs properly, use the boot script for multi-node multi-PU training.
- **VC_WORKER_HOSTS**, **VC_WORKER_NUM**, **VC_TASK_INDEX**, and **MA_NUM_GPUS** in **run.sh** for multi-node multi-PU training are environment variables preset in the ModelArts training container. For details about environment variables of a training container, see **Viewing Environment Variables of a Training Container**.
- **OUTPUT_PATH** in **run.sh** is the path for storing intermediate results such as model weights and training configurations during training. If **config.TRAIN.AUTO_RESUME** in the training script is set to **True** (default value), the latest model weight will be automatically loaded in the **OUTPUT_PATH** directory during training.

8. Save the code folder to OBS through obsutil and upload the code to the target directory of SFS through OBS.
9. In SFS, set the owner of the code file **Swin-Transformer-main** to **ma-user**.
   ```
   chown -R ma-user:ma-group Swin-Transformer
   ```
10. Run the following command to remove **\r** from the shell script:
    ```
    cd Swin-Transformer
    sed -i 's/\r//' run.sh
    ```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

> NOTE
>
> Shell scripts written in Windows have **\r\n** as the line ending, but Linux uses **\n** as the line ending. This means that Linux treats **\r** as part of the script and shows the error message "$'\r': command not found" when running it. To fix this, you need to remove **\r** from the shell script.

Debugging Code with Notebook

A notebook instance includes a **/cache** directory limited to 500 GB. Exceeding this limit causes the instance to restart. Since the ImageNet dataset surpasses 500 GB, consider using offline resources or a smaller portion for debugging within a notebook instance. (For details about debugging in a notebook instance, see **Debugging Code with Notebook**.)

Creating a Multi-Node Multi-PU Training Job

1. Log in to the **ModelArts console** and check whether access authorization has been configured for your account. For details, see **Configuring Agency Authorization for ModelArts with One Click**. If you have been authorized using access keys, clear the authorization and configure agency authorization.
2. In the navigation pane on the left, choose **Model Training** > **Training Jobs**. The training job list is displayed by default.
3. On the **Create Training Job** page, configure parameters and click **Submit**.
   - **Algorithm Type**: **Custom algorithm**
   - **Boot Mode**: **Custom image**
   - **Image**: Select the uploaded custom image.
   - **Boot Command**:
     ```
     cd /home/ma-user/work/code/Swin-Transformer && /home/ma-user/anaconda3/envs/pytorch/bin/pip install -r requirements.txt && /bin/sh run.sh
     ```
   - **Auto Restart**: Enable this function and set **Restarts**. In this way, when a node is faulty, the platform automatically restarts the job and isolates the faulty node.

     If this function is enabled, you need to configure the output path in the **run.sh** script. In this way, when a fault occurs, the training can be continued based on the model weight saved in the previous round, minimizing resource waste.
   - **Resource Pool**: In the **Dedicated Resource Pool** tab, select dedicated resource pool of the GPU flavor.
   - **Specifications**: Select required GPU specifications.
   - **Compute Nodes**: Select the number of required nodes.
   - **SFS Turbo**: Add the mount configuration and select the SFS name. The cloud mount path is **/home/ma-user/work**.

     > NOTE
     >
     > To ensure that the code path and boot command are the same as those for notebook debugging, set the cloud mount path to **/home/ma-user/work**.
4. Click **Submit**. On the information confirmation page, check the parameters, and click **OK**.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

5. Wait until the training job is created.

   After you submit the job creation request, the system will automatically perform operations on the backend, such as downloading the container image and code directory and running the boot command. A training job requires a certain period of time for running. The duration ranges from dozens of minutes to several hours, depending on the service logic and selected resources.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

8 Inference Deployment (New Version)

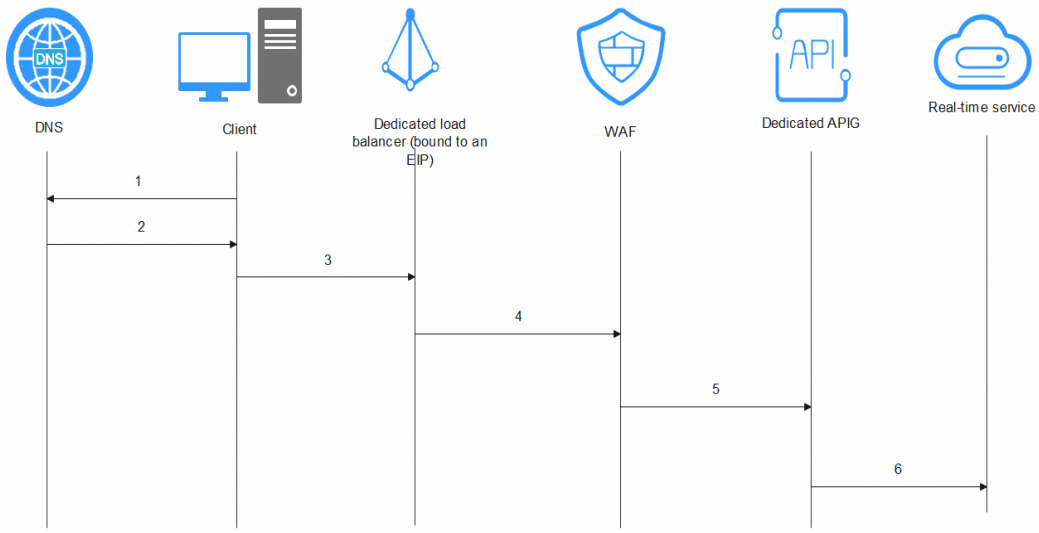
8.1 Accessing a Real-Time Service Through a Dedicated API Gateway, WAF, VPC, and ELB

Overview

This function is designed to ensure that public network users can securely and efficiently access ModelArts real-time services and flexibly integrate ModelArts prediction services.

This section describes how to use Dedicated API Gateway (APIG), Web Application Firewall (WAF), Virtual Private Cloud (VPC), and Elastic Load Balance (ELB) to access ModelArts real-time services over the intranet.

Figure 8-1 Real-time service access



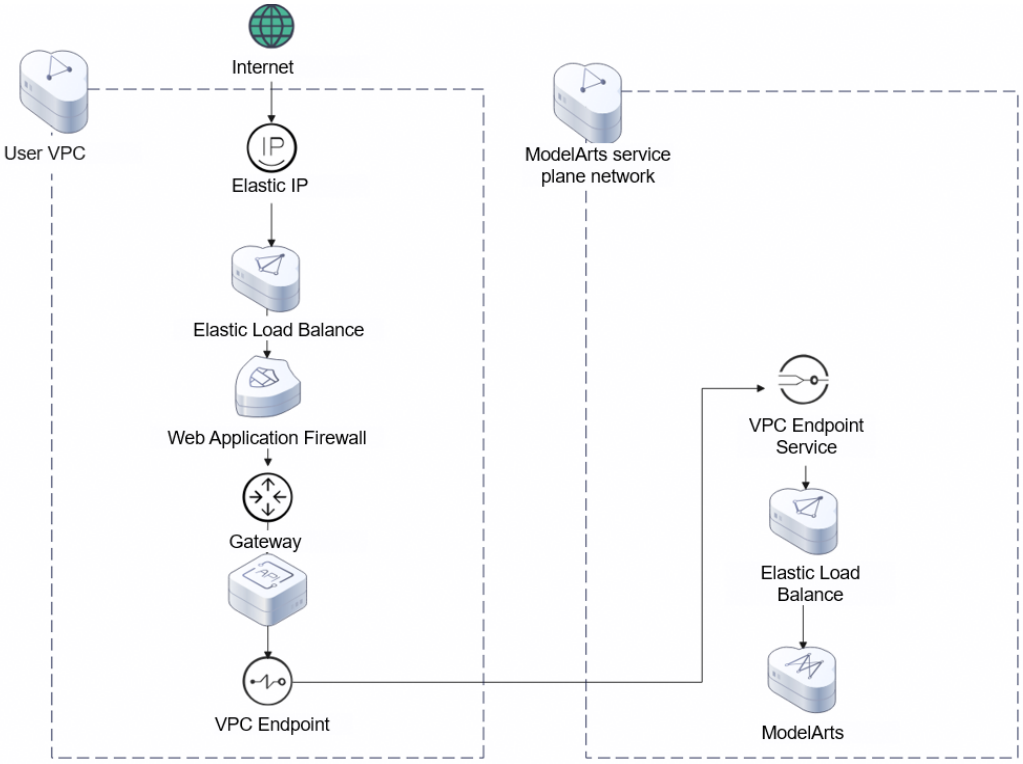
The details are as follows:

1. After a visitor enters a domain name in the browser, the client sends a request to the DNS service to query the domain name resolution address.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

2. DNS returns the domain name resolution address to the client.
3. The domain name resolution address returned by the DNS service is the EIP of the load balancer, and the client accesses the load balancer through the EIP.
4. The ELB load balancer forwards the traffic to WAF.
5. WAF checks the traffic, blocks abnormal traffic, and forwards normal traffic to the real-time service over the back-to-source IP address of the dedicated WAF engine.

Figure 8-2 Network architecture



Constraints

Constraints	Remarks	Workaround
Domain name	Prepare a public domain name for ICP filing in advance; the application and registration typically take three to five days.	Use the ELB IP address for access.
Response timeout	APIG limits the maximum response timeout of the backend service to 5 minutes.	If a longer timeout interval is required, you can remove APIG.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Resource and Cost Planning

This example uses these specifications only to set up access links. The QPS remains low. Final inference performance depends on each component's capabilities. Choose components that meet your service needs.

Resource	Specifications (for Reference)	Quantity
API Gateway (APIG)	• Billing mode: Pay-per-use • Basic edition • Single AZ: 50 Mbit/s • Dual-AZ or more: 100 Mbit/s • Private network connections per second: 1,000 For details about how to purchase specifications, see: **Specifications_Service Overview_API Gateway_Huawei Cloud**	1
Elastic IP (EIP)	• Billed by bandwidth • Bandwidth: 5 Mbit/s For details about how to purchase specifications, see: **Billing_Service Overview_Elastic IP_Huawei Cloud** **Lower Public Network Costs_Best Practices_Elastic IP_Huawei Cloud**	1
Elastic Load Balance (ELB)	• Network type: public network • Bandwidth: 5 Mbit/s For details about how to purchase specifications, see: **Load Balancer Selection and Service Planning_Best Practices_Elastic Load Balance_Huawei Cloud**	1

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Resource	Specifications (for Reference)	Quantity
Web Application Firewall (WAF)	• WI-100 • Estimated single instance performance • Throughput • 100Mbps • HTTP/HTTPS service • Recommended QPS: 1,000; maximum QPS: 2,000; WAF-to-Server persistent connections: 60,000 • Websocket • Concurrent connections: 1,000 For details about how to purchase specifications, see: **About Purchase and Specifications Change_Huawei Cloud** **How Do I Select Service QPS When Purchasing WAF?_Huawei Cloud**	1
VPC Endpoint	• Basic edition For details about how to purchase specifications, see: **Lower Network Costs_Best Practices_Virtual Private Cloud_Huawei Cloud** **Bandwidth_FAQs_Virtual Private Cloud_Huawei Cloud**	1

Permissions

Table 8-1 Permissions required for creating components

Component	Component Creation Permission	Official Website
VPC	VPC FullAccess or VPC Administrator	**Permissions_Service Overview_Virtual Private Cloud_Huawei Cloud**
EIP	VPC FullAccess or VPC Administrator	**EIP Permissions_Service Overview_Elastic IP_Huawei Cloud**

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Component	Component Creation Permission	Official Website
APIG	• System role permissions: You can create an instance only if you have both the APIG Administrator and VPC Administrator permissions. • System policy: You only need to have the APIG FullAccess permission.	Quickly Opening and Calling APIs_Getting Started_API Gateway_Huawei Cloud
ELB	ELBFullAccess or ELB Administrator	Permissions_Service Overview_Elastic Load Balance_Huawei Cloud
WAF	WAF Administrator or WAF FullAccess	Permission Dependency of the WAF Console_Permissions Management_User Guide_Web Application Firewall_Huawei Cloud

When using the private network connection function, you must have the following permissions:

Table 8-2 Permissions required for private network connection

Permission	action	Scenario
Obtaining details about a workspace	modelarts:workspace:get	Accessing workspaces
Obtaining model services	modelarts:service:list	Obtaining services
Obtaining dedicated resource pools	modelarts:pool:list	Obtaining resource pools
Obtaining VPCs	vpc:vpcs:list	Private network connection page and private network connection request creation

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Permission	action	Scenario
Obtaining subnets or subnet details	vpc:subnets:get	Private network connection page and private network connection request creation
Updating a service	modelarts:service:update	Creating, managing, and deleting private network connections at the service level
Obtaining private network connections	iam:modelarts:intranetConnection:list	Querying the private network connection list
Modifying private network connections	iam:modelarts:intranetConnection:update	Modifying private network connections
Creating private network connections	iam:modelarts:intranetConnection:create	Creating private network connections
Deleting private network connections	modelarts:intranetConnection:delete	Deleting private network connections

You also need to grant the agency permissions below to ModelArts. For details about how to add ModelArts authorization, see **Configuring Agency Authorization for ModelArts with One Click**.

Table 8-3 Permissions granted to ModelArts for private network connection

Permission	action	Scenario
Creating endpoints	vpcep:endpoints:create	Creating and changing private network connections
Deleting endpoints	vpcep:endpoints:delete	Deleting private network connections
Querying endpoint details	vpcep:endpoints:get	Creating and changing private network connections

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Permission	action	Scenario
Querying endpoints	vpcep:endpoints:list	Creating and changing private network connections

Process

Step	Description
Step 1: Creating a VPC	Shows how to create a virtual private cloud (VPC).
Step 2: Creating a Private Network Connection	Shows how to create a private network connection for a service.
Step 3: Purchasing an API Gateway	Shows how to purchase an API gateway.
Step 4: Creating an API Through APIG	Shows how to create an API.
Step 5: Creating an Elastic Load Balancer	Shows how to create a load balancer.
Step 6: Adding a Listener	Shows how to add a listener and connect the load balancer to WAF.
Step 7: Using WAF to Protect APIG	Shows how to connect WAF to APIG.
Step 8: Accessing a Real-Time Service from a Public Network	Shows how to access real-time inference services from the public network.

Step 1: Creating a VPC

1. On the **Create VPC** page, set the VPC and subnet parameters as prompted. For more information, see **Creating a VPC with a Subnet**.
2. Click **Create Now**.

Step 2: Creating a Private Network Connection

1. Log in to the **ModelArts console**. In the navigation pane, choose **Model Inference** > **Real-Time Inference**.
2. Click **Private Network Connections**.
3. In the **My Requests** tab, click **Create**.
4. Set parameters and click **OK**.

   Check the request status in the **My Requests** tab. If the request status is **Connection succeeded**, the request is approved.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

5. Obtain **Access Address** from **My Requests**.

Table 8-4 Parameters for private network connection request

Parameter	Description	Recommended Value
Service ID	ID of the inference service that needs to be accessed through the private network.	You have obtained the real-time service ID on the service details page by clicking the service name in the inference service list.
Network Type	Type of the network for which the private network is enabled. • **User network**: Connect the private network between your VPC and inference service. • **Resource pool network**: Connect the private network between your resource pool and inference service.	**User network**.
VPC	When you select **User network**, choose a VPC for private network connection.	Select the VPC created in **Step 1: Creating a VPC**.
Subnet	When you select **User network**, choose a subnet for private network connection.	Select the subnet created in **Step 1: Creating a VPC**.
Resource Pool	When you select **Resource pool network**, choose a resource pool for private network connection.	N/A.
Custom Access Address	Allow users to access a real-time service through a load balancer using a public domain name.	Domain name provided by the user.

Step 3: Purchasing an API Gateway

To manage real-time services, you need to purchase a dedicated gateway instance, create an API, and bind a domain name to the API.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

1. On the **Buy Gateway** page, configure the instance information. For more parameter descriptions, see **Creating a Gateway**.

   Purchase different instances based on service requirements. The final price is subject to the price displayed on the page.

   In this example, the basic edition is used as an example. You can ignore the public network inbound and outbound ports. Select the VPC and subnet created in **Step 1: Creating a VPC**, and open ports 80 and 443 in the inbound direction of the security group. For more parameter descriptions, see **Creating a Gateway**.
2. Click **Next**.
3. Confirm the gateway configurations, read and confirm your acceptance of the service agreement, and click **Pay Now**. The gateway is created with the status displayed on the screen.

Step 4: Creating an API Through APIG

1. Obtain the private network connection address.
   a. Log in to the **ModelArts console**. In the navigation pane, choose **Model Inference** > **Real-Time Inference**.
   b. Click **Private Network Connections**.
   c. Click **My Requests**, find the private network connection created in **Step 2: Creating a Private Network Connection**, and copy the value in the **Access Address** column. Use **https://1xx.xxx.xxx.xxx/v2/infer/xxx-xxxx-xxxxxxx** as an example to create an API.

      Note: The access address is only available when the status is **Connection succeeded**.
2. Create a load balance channel for the API.
   a. Log in to the APIG console, select the APIG instance created in **Step 3: Purchasing an API Gateway** from the drop-down list in the upper left corner, and choose **API Management** > **API Policies** in the navigation pane on the left.
   b. Click **Load Balance Channels**.
   c. Click **Create Load Balance Channel** and configure basic information. **Table 8-5** describes some parameters.

      For details about more parameters, see **(Optional) Creating a Load Balance Channel**.

      Table 8-5 Parameters for creating a load balance channel

Parameter	Description	Recommended Value
Name	Select a billing mode based on your service requirements.	VPC_xxxx

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Parameter	Description	Recommended Value
Port	Set this parameter based on the private network connection address. HTTP: 80 HTTPS: 443	443
Routing Algorithm	The algorithm to be used to forward requests to cloud servers you select. The following routing algorithms are available: • **WRR**: Requests are forwarded to each cloud server in order of server weight. • **WLC**: Requests are forwarded to the cloud server with the fewest live connections in order of server weight. • **SH**: Requests are forwarded to the cloud server by source IP address. Requests with the same source IP address are forwarded to the same server unless the server is unavailable. • **URI hashing**: Requests are forwarded to the cloud server by request path. Requests from the same path are forwarded to the same server unless the server is unavailable.	Weighted round robin
Type	• **Server**: API requests will be distributed to ECSs or specified server IP addresses in the channel. • **Microservice**: API requests will be distributed to microservice IP addresses in the channel. • **Reference:** you can reference other existing load balance channels.	**Server**
Mode	Configure servers if **Type** is set to **Server**.	**Specify IP addresses**

d. After entering the basic information, click **Create Server Group**. In the displayed dialog box, enter the server group information and click **OK**.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

For details about more parameters, see **(Optional) Creating a Load Balance Channel**.

Table 8-6 Parameters for creating a server group

Parameter	Description	Value
Group Name	Enter a server group name. Using naming rules facilitates future search. It can contain 3 to 64 characters, starting with a letter. Only letters, digits, underscores (_), hyphens (-), and periods (.) are allowed.	**service-group-1**
Weight	Enter the weight of the server group. The larger the weight, the more requests can be forwarded to the servers in the group. The default value is **1**. The value ranges from **0** to **100**.	**1**
Description	Enter a brief description of the server group. Enter 1 to 255 characters.	-

e. Click **Add Backend Server Address** and enter a backend server address based on the following table.

Parameter	Description	Value
Backend Server Address	IP address of the private network connection address, for example, https://192.xxx.xxx.122. /v2/infer/xxx-xxxx-xxxxxxx: Enter 192.xxx.xxx.122.	192.xxx.xxx.122

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Parameter	Description	Value
Port	Enter the following information based on the request protocol of the private network connection address: HTTP: 80 HTTPS: 443	443

f. Click **Finish**.

3. Create an API.

a. Log in to the APIG console, select the APIG instance created in **Step 3: Purchasing an API Gateway** from the drop-down list in the upper left corner, and choose **API Management** > **APIs** in the navigation pane on the left.

b. In the upper right corner of the page, click **Create API**. Configure the parameters based on the following table. **Table 8-7** describes some parameters. For details about more parameters, see **Creating a REST API**.

Table 8-7 API parameters

Parameter	Description	Value
API Name	Enter 3 to 255 characters, starting with a letter or digit. Use only letters, digits, and these special characters: -_./:()	API_xxxx
Group	Select or create an API group.	DEFAULT
URL	Example: https://192.xxx.xxx.122 /v2/infer/xxx-xxxx-xxxxxxx provides the POST /v1/sse interface. Request method: Select the method that matches the API. Request protocol: Select the protocol that matches the API. Path: The request path can include parameters enclosed with braces ({}) (example: /a/{b}). For prefix match, add a plus sign (+) to the end of a parameter (example: /a/{b+}).	POST HTTPS /v2/infer/xxxxx/v1/sse
Gateway Response	default	default

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Parameter	Description	Value
Type	Public APIs can be published in the RELEASE environment and sold. Private APIs cannot be published for sale.	Public
Authentication Mode	App authentication: Use AppKey & AppSecret authentication. IAM authentication: Only authorized users can access the API. Custom authentication: Custom authentication is used to control API access. No authentication: All users can access the API. This option is not recommended. Note: If the request includes valid authentication details, choose **None** to skip repeated verification by API Gateway, boosting API performance.	**None**

c. Click **Next**. The backend configuration page is displayed.

Table 8-8 describes some parameters. For details about more parameters, see **Creating a REST API**.

Table 8-8 Backend configuration parameters

Parameter	Description	Value
Backend Type	Select the backend service type.	HTTP&HTTPS
Load Balancing	Select **Single server group**.	DEFAULT
URL	Example link *https://192.xxx.xxx.122/v2/infer/xxx-xxxx-xxxxxxx* provides the POST /v1/sse API. Request method: Select the method that matches the API. Request protocol: Select the protocol that matches the API. Load balance channel: Select the channel created earlier. Path: The request path can include parameters enclosed with braces ({}) (example: /a/{b}). For prefix match, add a plus sign (+) to the end of a parameter (example: /a/{b+}).	POST HTTPS VPC_xxxx /v2/infer/xxxxxx/v1/sse

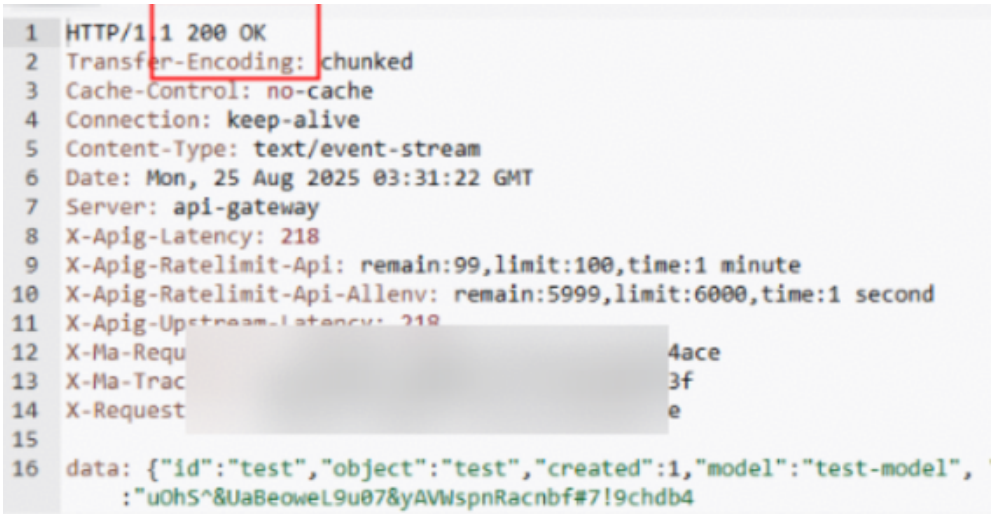
 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Parameter	Description	Value
Timeout (ms)	API Gateway limits the maximum response time to 5 minutes. If a longer response time is required, the request cannot be sent to API Gateway. Unit: millisecond.	5000

d. Click **Finish**. Wait until the API debugging page is displayed.

4. On the API debugging page, click **Debug** in the upper right corner and use the default domain name provided by API Gateway. If the API responds with 200, the API is configured successfully.

Figure 8-3 API debugging



5. After the API is debugged, click **Publish Latest Version** in the upper right corner. In the displayed dialog box, select an environment and click **OK**.

   In this case, select **RELEASE**.

6. Bind a domain name. In the navigation pane on the left, choose **API Management** > **API Groups**, and click the name of the API group to which the API belongs. The details page is displayed.

   Click **Bind Independent Domain Name**, enter the prepared independent domain name, and click **OK**.

Step 5: Creating an Elastic Load Balancer

1. Log in to the **Elastic Load Balance console** and click **Buy Elastic Load Balancer**.
2. On the **Buy Elastic Load Balancer** page, set the parameters as prompted. For details about the parameters to be configured, see **Table 8-9**. Configure other parameters as required. For details about other parameters, see **Creating a Dedicated Load Balancer**.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Table 8-9 ELB parameters

Parameter	Value
Network Type	Private IPv4 network.
VPC	Select the VPC created in **Step 1: Creating a VPC**.
Frontend Subnet	Select the subnet created in **Step 1: Creating a VPC**.
IPv4 Address	Automatically-assigned IP address.
Backend Subnet	Subnet of the load balancer.
IP as a Backend	Enabled.

Step 6: Adding a Listener

1. On the **load balancer** details page, click the **Listeners** tab and click **Add Listener**.
2. On the **Add Listener** page, set the parameters according to **Table 8-10**. You can configure other parameters as required. For details about more parameters, see **Adding an HTTPS Listener**.

   Table 8-10 Parameters for adding a listener

Parameter	Value
Frontend Protocol	Set this parameter based on the protocol of the service call API of the real-time service. For example, if the service call API of the real-time service is **HTTP://{host}:8080**, set this parameter to **HTTP**.
Listening Port	If **Frontend Protocol** is **HTTP**, set this parameter to **80**. If **Frontend Protocol** is set to **HTTPS**, set this parameter to **443**.

3. Click **Next: Configure Request Routing Policy** and set the parameters. For details, see **Table 8-11**. For more information, see **Creating a Backend Server Group**.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Table 8-11 Parameters for configuring a backend server group

Parameter	Description	Value
Backend Server Group	Create a backend server group by referring to **Creating a Backend Server Group**.	**Create new**
Name	Specifies the name of the backend server group.	server_group-best-0829
Backend Server Group Type	Specifies the type of the backend server group. • **Hybrid**: You can add cloud servers and supplementary network interfaces as backend servers, and add IP addresses as backend servers when **IP as a Backend** is enabled. When you create a hybrid backend server group, you must specify a VPC and associate the backend server group with a load balancer in this VPC. • IP as a backend server: Only IP addresses can be added as backend servers. You can add IP addresses as backend servers only when **IP as a Backend** is enabled.	Hybrid
Backend Protocol	Determined by the private network connection in **Step 2: Creating a Private Network Connection**. For example, if the private network connection is *https://1xx.xxx.xx.xxx:443/v2/infer/xxxx*, select **HTTPS**.	HTTPS

4. Click **Next: Add Backend Server** in the lower right corner. In this case, you do not need to perform any operation on this page.
5. Click **Next: Confirm**, confirm the settings, and click **Submit**.

   After the listener is created, the listener list page is displayed.

Step 7: Using WAF to Protect APIG

1. **Log in to the WAF console**. In the upper right corner of the page, click **Buy WAF**.
2. On the **Buy Web Application Firewall** page, select **Dedicated Mode** for **WAF Mode**.
3. Set other parameters as required.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

In this example, the VPC, subnet, and security group must be those created in **Step 1: Creating a VPC**. For more parameter description, see **Buying Dedicated Mode WAF**.

4. Confirm the parameters and click **Buy Now**.
5. On the **Confirm Configuration** page, confirm the order details, check the boxes before "I have read and agree to the WAF Disclaimer." and "I agree to assign permissions of the following roles to WAF", and click **Pay**.
6. After the purchase is successful, click **Back to Dedicated Engine List** to return to the homepage.
7. Add to ELB.
   a. On the WAF console, choose **Instance Management** > **Dedicated Engine** in the navigation pane on the left. In the dedicated engine list, locate the newly created WAF instance and choose **More** > **Add to ELB** in the **Operation** column.
   b. On the page that appears, select the ELB created in **Step 5: Creating an Elastic Load Balancer**, the listener created in **Step 6: Adding a Listener**, and the backend server group. Click **Next**.
   c. On the **Set Ports and Weights** page, enter the service port number based on the protocol. For HTTP requests, enter **80**. For HTTPS requests, enter **443**. Then, click **OK**.
   d. Return to the **ELB page** in **Step 5: Creating an Elastic Load Balancer**, find the corresponding load balancer, and click the listener to go to the basic information page.
   e. In the backend server group area on the basic information page, click the backend server group name.
   f. On the page that appears, click the **Backend Servers** tab, expand the cloud server drop-down list, and check whether the private IP address is the IP address of the WAF instance.
8. In the navigation pane on the left of the WAF console, choose **Website Settings**.
9. In the upper left corner of the website list, click **Add Website**, select **Dedicated Mode**, and click **Configure Now**.
10. In the **Add Website** dialog box, set the parameters described in **Table 8-12**, and set other parameters as required. For more information about the parameters, see **Add a Website to WAF**.

Table 8-12 Adding a website

Parameter	Description	Value
Protected Object	The domain name of the website you want to protect. It can be a single domain name or a wildcard domain name. Enter the prepared domain name.	www.xxxx.com

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

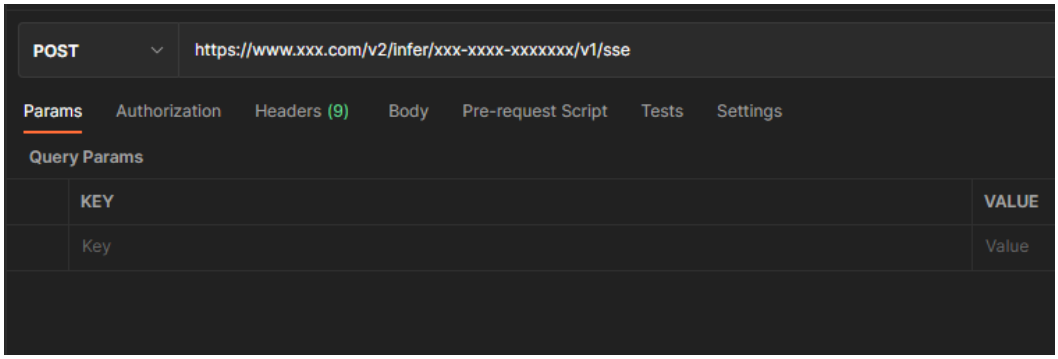
Parameter	Description	Value
Server Configuration	Enter the port number and protocol based on the private network connection address. HTTP: 80. HTTPS: 443. VPC: Select the VPC created in **Step 1: Creating a VPC**. Private IP address: APIG IP address. Go to the APIG page and click the created APIG. On the details page, find the VPC access address.	HTTPS HTTPS vpc-xxxx 443 *192.xxx.xxx.xxx* (private APIG IP address)
Layer-7 Proxy Used	Layer-7 proxy: Web proxy products for layer-7 request forwarding are used, products such as anti-DDoS, CDN, and other cloud acceleration services. Note: If a layer-7 proxy is configured, WAF reads the real client IP address from the related fields in the header.	No

11. Click **OK**. The list page is displayed. Click the domain name added for **Protected Object**. The detailed configuration page is displayed.
12. Check whether WAF is normal. For details, see **Test Dedicated WAF Instances**.

Step 8: Accessing a Real-Time Service from a Public Network

Use a browser or Postman to access the real-time service. If security authentication is configured for the API, the security authentication must be added to the request. In this example, no authentication is used.

Figure 8-4 Accessing a real-time service through a private network



 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Table 8-13 Access address description

Address	Description
www.address.com	Protected domain name added in **Step 7: Using WAF to Protect APIG**.
/v2/infer/xxx-xxxx-xxxxxxx	API created in **Step 4: Creating an API Through APIG**.

8.2 Accessing a Real-Time Service Through ELB

Context

ELB automatically distributes incoming traffic across multiple backend servers based on configured listening rules. It expands the service capabilities of your applications and improves their availability by eliminating individual faults.

ModelArts allows public network users to securely and quickly access its real-time services and easily integrate prediction services. It uses ELB for private network access. ELB handles incoming traffic and sends it to the backend inference service instances based on the forwarding rules.

Permissions

When using the ELB access function, you must have the following permissions.

Table 8-14 User permissions

Permission	Action	Scenario
Obtaining workspace details	modelarts:workspace:get	Accessing workspaces
Obtaining model services	modelarts:service:list	Obtaining services
Updating services	modelarts:service:update	Creating, modifying, or deleting ELB accesses
Deleting services	modelarts:service:delete	Deleting ELB accesses
Obtaining dedicated resource pools	modelarts:pool:list	Obtaining resource pools
Obtaining VPCs	vpc:vpcs:list	Creating or modifying ELB accesses

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Permission	Action	Scenario
Obtaining subnet information	vpc:subnets:get	Creating or modifying ELB accesses
Obtaining load balancers	elb:loadbalancers:list	Creating or modifying ELB accesses
Obtaining certificates	elb:certificates:list	Creating or modifying ELB accesses
Obtaining listeners	elb:listeners:list	Creating or modifying ELB accesses

You also need to grant the agency permissions below to ModelArts. For details about how to add ModelArts authorization, see **Configuring Agency Authorization for ModelArts with One Click**.

Table 8-15 Agency permissions granted by you

Permission	Action	Scenario
Creating endpoints	vpcep:endpoints:create	Creating or modifying ELB accesses
Deleting endpoints	vpcep:endpoints:delete	Modifying or deleting ELB accesses
Obtaining endpoint details	vpcep:endpoints:get	Creating or modifying ELB accesses
Obtaining endpoints	vpcep:endpoints:list	Creating or modifying ELB accesses
Obtaining load balancers	elb:loadbalancers:get	Creating or obtaining ELB accesses
Obtaining load balancers	elb:loadbalancers:list	Obtaining ELBs and ELB accesses
Creating listeners	elb:listeners:create	Creating or modifying ELB accesses
Obtaining listeners	elb:listeners:list	Obtaining ELB accesses
Obtaining listener details	elb:listeners:get	Modifying, deleting, or obtaining ELB accesses

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Permission	Action	Scenario
Deleting listeners	elb:listeners:delete	Modifying or deleting ELB accesses
Updating listeners	elb:listeners:put	Creating or modifying ELB accesses
Obtaining forwarding policies	elb:l7policies:list	Creating or modifying ELB accesses
Creating forwarding policies	elb:l7policies:create	Creating or modifying ELB accesses
Deleting forwarding policies	elb:l7policies:delete	Modifying or deleting ELB accesses
Creating backend server groups	elb:pools:create	Creating or modifying ELB accesses
Obtaining backend server group details	elb:pools:get	Modifying or deleting ELB accesses
Deleting backend server groups	elb:pools:delete	Modifying or deleting ELB accesses
Obtaining backend server groups	elb:pools:list	Deleting ELB accesses
Creating backend server group members	elb:members:create	Creating or modifying ELB accesses
Obtaining backend server group members	elb:members:list	Modifying or deleting ELB accesses
Obtaining backend server group member details	elb:members:get	Modifying or deleting ELB accesses
Deleting backend server group members	elb:members:delete	Modifying or deleting ELB accesses
Obtaining certificates	elb:certificates:list	Creating or modifying ELB accesses
Obtaining forwarding rule details	elb:l7rules:get	Creating or modifying ELB accesses

Billing

Using ELB and VPC endpoints is billed. For details, see **Elastic Load Balance Billing** and **VPC Endpoint Billing**.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Step 1: Creating an ELB

You have created a dedicated load balancer. For details, see **Creating a Dedicated Load Balancer**. The following table lists the requirements for creating a load balancer.

Table 8-16 Parameters for creating a load balancer

Parameter	Description
Specifications	Select a dedicated load balancer.
Load Balancing Type	Select **Application** or **Network + Application**.
Network Configuration	Select **IP as Backend Servers**.

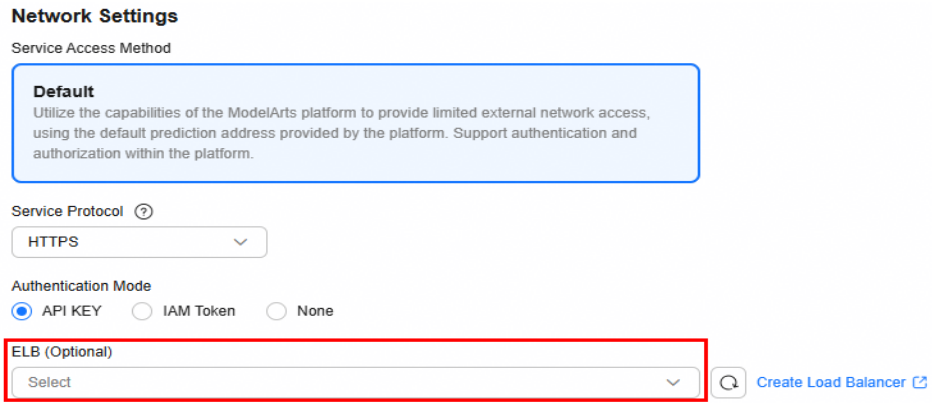
Step 2: Configuring ELB Access in a ModelArts Real-Time Inference Service

When configuring the inference service information, configure the ELB parameters. For details, see **Configuring Service Information**.

The ELB parameters are as follows:

1. Log in to the **ModelArts console**. In the navigation pane, choose **Model Inference** > **Real-Time Inference**.
2. Click **Deploy**. The **Deploy Real-Time Service** page is displayed. When configuring the inference service information, select the created load balancer under **ELB (Optional)**.

   Figure 8-5 Selecting a load balancer



3. After selecting a load balancer, if the service uses HTTPS or WSS, configure the SSL parsing mode. Choose either unidirectional or bidirectional authentication. For unidirectional authentication, configure only the server certificate. For bidirectional authentication, configure both the server and CA certificates. In addition, you can configure SNI.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Figure 8-6 One-way authentication

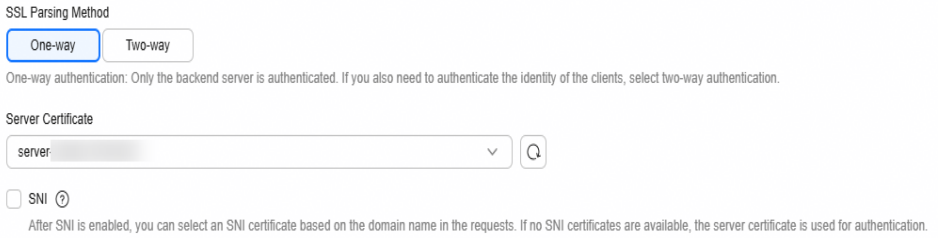
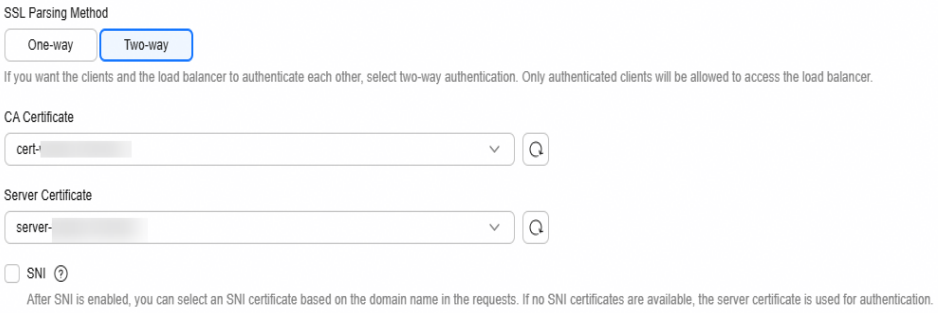


Figure 8-7 Two-way authentication



The CA certificate, server certificate, and SNI certificate must be created in advance. For details, see **Adding a Certificate**. For more information about SNI, see **Using SNI Certificates for Access Through Multiple Domain Names**.

For details about other parameters of service information and deployment configuration, see **Configuring Service Information** and **Deploying a Real-Time Inference Service Using a Single Node**. After you finish the deployment setup, follow these steps to access and use the service.

Step 3: Accessing the Real-Time Service Using the ELB Address

After configuring an ELB for the service and deploying the service, you can click the service in the real-time service list or on the service details page to view the ELB access address.

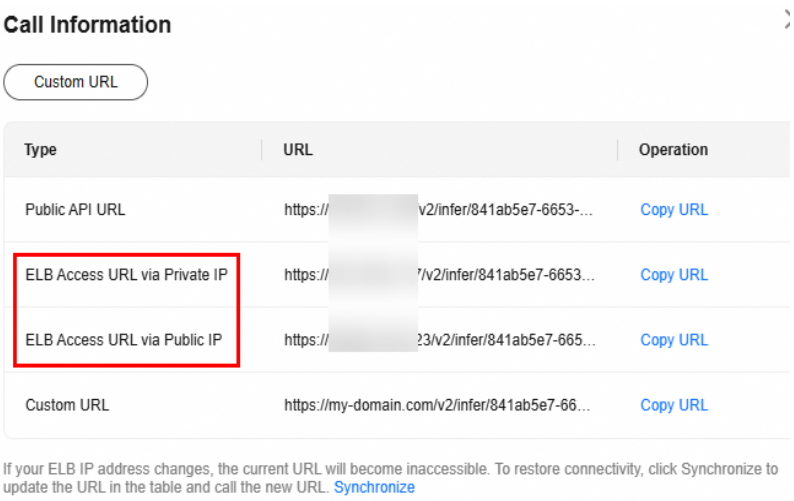
Figure 8-8 Calling a service



The service call information will display the private or public network call URLs one by one based on the IP addresses of the ELB.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Figure 8-9 Calling information

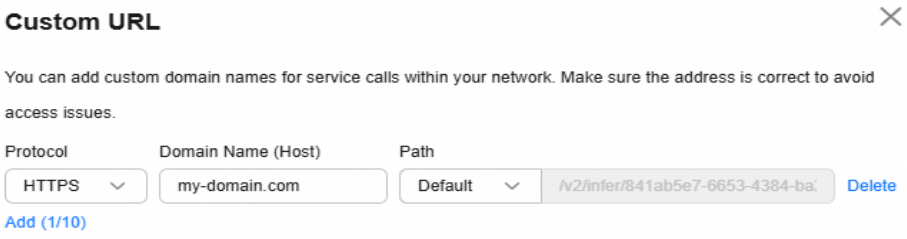


The format of the ELB call address for real-time services is: *${service_protocol}*://*${ELB_instance_IP}*/**v2/infer/***${real-time_service_ID}*/*${specific_API}*. For example: **http://10.10.xx.xxx/v2/infer/d64cda4f-af92-4bd3-804d-xxxxx/v1/chat/completions**.

Since users can change the private IP address or EIP bound to an ELB instance at any time via the ELB console or ELB APIs, a **Synchronize** button is provided at the bottom of the real-time service call information section. This allows one-click synchronization of the latest ELB IP address to ensure that the real-time service remains accessible via the new IP address.

If you have already set up other network components that handle Layer-4 traffic before deploying the real-time service, click **Custom URL** in the **Call Information** and add the IP address or domain name of the network component. This allows you to access the real-time service using the IP address or domain name. For example, if you have also connected APIG before configuring the ELB and wants to access the real-time service via the APIG domain name, you need to add the APIG domain name to the custom addresses.

Figure 8-10 Modifying the access address



VPC Endpoint Creation Rules

If the VPC associated with the ELB accessed by you is connecting for the first time, a VPC endpoint will be automatically created under this VPC to bridge the network between this VPC and the real-time service. The IP address of the VPC endpoint will be used as the backend server of the ELB.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

If the VPC endpoint automatically created by the ModelArts real-time service is no longer used, it will be automatically deleted.

ELB Resources Creation Rules

When you access an ELB, a series of related resources will be created under the ELB, including listeners, backend server groups, and forwarding policies. Specifically:

- If the service protocol is HTTP, the ELB creates an HTTP listener on port 80. If the service protocol is HTTPS, the ELB creates an HTTPS listener on port 443.
- If the service runs in synchronous mode, a rule with the path **/v2/infer** forwards calls to the synchronous real-time service. If the service runs in asynchronous mode, a rule with the path **/v2/async-infer** forwards calls to the asynchronous real-time service.

If the above ELB-related resources already exist, they will be reused; if they do not exist, they will be created automatically. During the reuse process, configuration conflicts may occur. Common configuration conflicts are as follows:

1. Listeners: If the service protocol is **HTTP**, but the ELB is already bound to a listener with a different protocol (e.g., HTTPS, TCP) using port 80, it is considered a configuration conflict and ELB access will fail.
2. Forwarding policies: If the service runs in synchronous mode, and the listener already contains a forwarding rule with a path prefix matching **/v2/infer** whose forwarding target is not a backend server group, or whose backend server in the group is not the target VPC endpoint IP, it is considered a configuration conflict and ELB access will fail.

If an ELB automatically created by the ModelArts real-time service is no longer in use, the aforementioned ELB-related resources will be automatically cleaned up.

Checking the ELB Interconnection Status

On the inference service details page, view the ELB interconnection status in the network configuration area. If the ELB fails to be connected or disconnected, you can retry on the inference details page.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Figure 8-11 Checking the ELB interconnection status

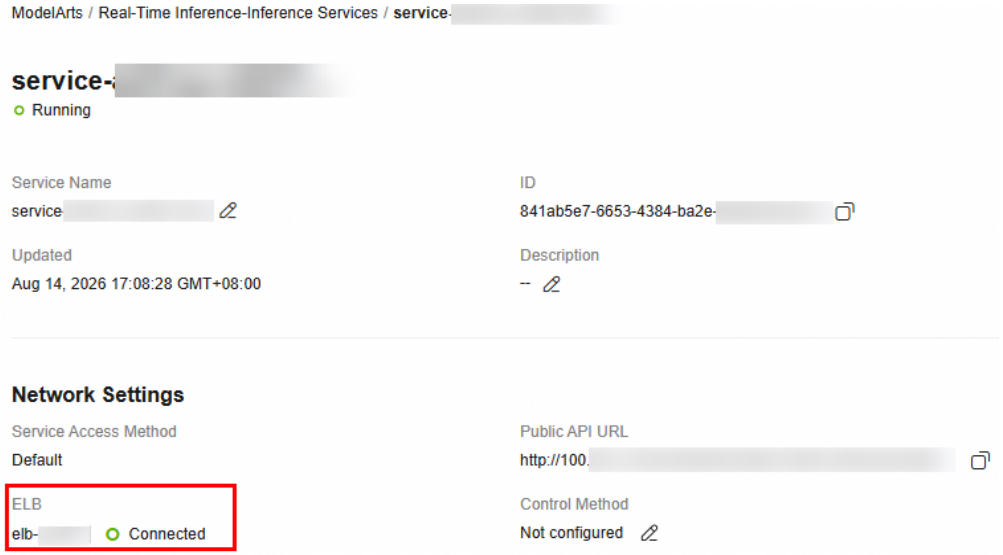


Table 8-17 ELB interconnection status

Status	Description	Possible Cause	Constraint
Connecting	Creating ELB and VPCEP-related resources and establishing network connectivity between ELB and the real-time service.	N/A	Cannot switch to another ELB or disconnect ELB midway.
Connected	Network connectivity between ELB and the real-time service is established. The real-time service can now be accessed via ELB.	N/A	None
Connection failed	Failed to establish network connectivity between ELB and the real-time service.	Lack of relevant agency permissions, such as **elb:listeners:create**.	Cannot switch to another ELB or disconnect ELB midway.
Disconnecting	Cleaning up ELB and VPCEP-related resources.	N/A	Cannot switch to another ELB midway.
Disconnection failed	Failed to clean up ELB or VPCEP-related resources.	Lack of relevant agency permissions, such as **elb:listeners:delete**.	Cannot switch to another ELB midway.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

9 Inference Deployment (Old Version)

9.1 Creating a Model Using a Custom Engine

When using a custom engine to create a model, you can select your image stored in SWR as the engine and specify a file directory in OBS as the model package. In this way, bring-your-own images can be used to meet your dedicated requirements.

Specifications for Using a Custom Engine to Create a Model

To use a custom engine to create a model, ensure the SWR image, OBS model package, and file size comply with the following requirements:

- SWR image specifications
  - A common user named **ma-user** in group **ma-group** must be built in the SWR image. Additionally, the UID and GID of the user must be 1000 and 100, respectively. The following is the dockerfile command for the built-in user:
    ```
    groupadd -g 100 ma-group && useradd -d /home/ma-user -m -u 1000 -g 100 -s /bin/bash ma-user
    ```
  - Specify a command for starting the image. In the dockerfile, specify **cmd**. The following shows an example:
    ```
    CMD sh /home/mind/run.sh
    ```
    Customize the startup entry file **run.sh**. The following is an example.
    ```
    #!/bin/bash

    # User-defined script content
    ...

    # run.sh calls app.py to start the server. For details about app.py, see "HTTPS Example".
    python app.py
    ```
    NOTE

    You can also customize the boot command for starting an image. Enter the customized command during model creation.
  - The service can use the HTTPS/HTTP protocol and listening container port. The port and protocol can be set based on the site requirements. The default request protocol and port number provided by ModelArts are HTTPS and 8080, respectively. For details, see the **HTTPS example**.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

  - (Optional) The health check URL must be **/health**.

- OBS model package specifications

  The name of the model package must be **model**. For details about model package specifications, see **Model Package Structure**.

- File size specifications

  When a public resource pool is used, the total size of the downloaded SWR image (not the compressed image displayed on the SWR page) and the OBS model package cannot exceed 30 GB.

HTTPS Example

Use Flask to start HTTPS. The following is an example of the web server code:

```
from flask import Flask, request
import json

app = Flask(__name__)

@app.route('/greet', methods=['POST'])
def say_hello_func():
    print("----------- in hello func ----------")
    data = json.loads(request.get_data(as_text=True))
    print(data)
    username = data['name']
    rsp_msg = 'Hello, {}!'.format(username)
    return json.dumps({"response":rsp_msg}, indent=4)

@app.route('/goodbye', methods=['GET'])
def say_goodbye_func():
    print("----------- in goodbye func ----------")
    return '\nGoodbye!\n'


@app.route('/', methods=['POST'])
def default_func():
    print("----------- in default func ----------")
    data = json.loads(request.get_data(as_text=True))
    return '\n called default func !\n {} \n'.format(str(data))

@app.route('/health', methods=['GET'])
def healthy():
    return "{\"status\": \"OK\"}"

# host must be "0.0.0.0", port must be 8080
if __name__ == '__main__':
    app.run(host="0.0.0.0", port=8080, ssl_context='adhoc')
```

Debugging on a Local Computer

Perform the following operations on a local computer with Docker installed to check whether a custom engine complies with specifications:

1. Download the custom image, for example, **custom_engine:v1** to the local computer.
2. Copy the model package folder **model** to the local host.
3. Run the following command in the same directory as the model package folder to start the service:

   ```
   docker run --user 1000:100 -p 8080:8080 -v model:/home/mind/model    custom_engine:v1
   ```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

> **NOTE**
>
> This command is used for simulation only because the directory mounted to **-v** is assigned the root permission. In the cloud environment, after the model file is downloaded from OBS to **/home/mind/model**, the file owner will be changed to **ma-user**.

4. Start another terminal on the local computer and run the following command to obtain the expected inference result:

```
curl https://127.0.0.1:8080/${Request path to the inference service}
```

Deployment Example

The following section describes how to use a custom engine to create a model.

1. Create a model and view model details.

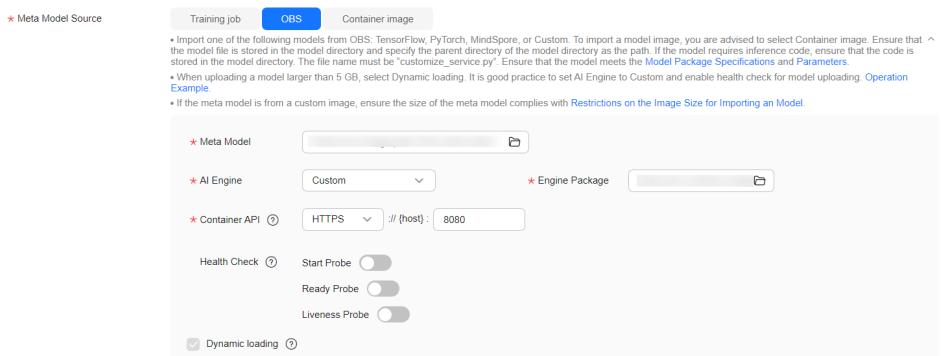
   Log in to the **ModelArts console**. Choose **Model Management** and click **Create Model**. On the page that is displayed, set the following parameters:

   - **Meta Model Source**: Choose **OBS**.
   - **Meta Model**: Select a model package selected from OBS
   - **AI Engine**: Choose **Custom**.
   - **Engine Package**: Select an SWR image.
   - **Container API**: Set the port and protocol based on the actual image usage.

     Retain the default settings for other parameters.

   Click **Create now**. Wait until the model status changes to **Normal**.

   Figure 9-1 Creating a model

   Click the model name to access its details page.

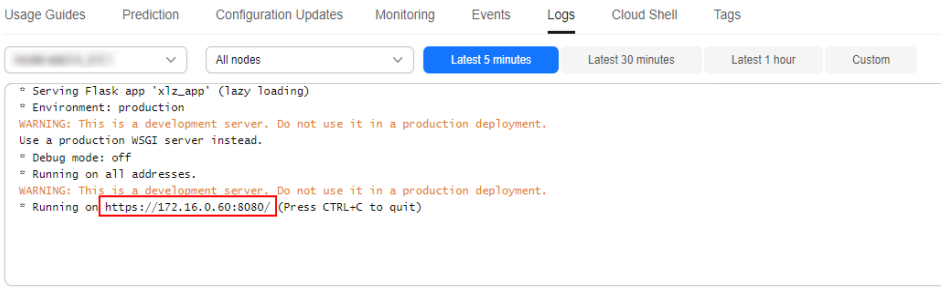
2. Deploy the AI application as a service and view service details.

   On the model details page, choose **Deploy** > **Real-Time Services** in the upper right corner. On the **Deploy** page, select a proper instance flavor (for example, **CPU: 2 vCPUs 8 GB**), retain default settings for other parameters, and click **Next**. When the service status changes to **Running**, the service has been deployed.

   Click the service name. On the page that is displayed, view the service details. Click the **Logs** tab to view the service logs.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

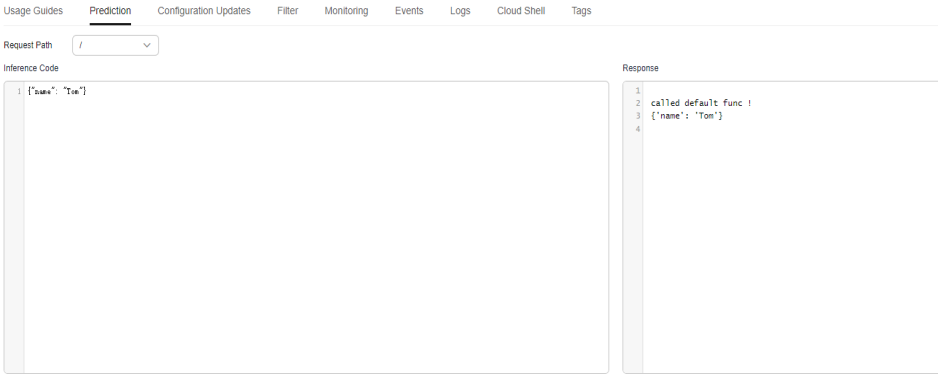
Figure 9-2 Logs



3. Use the service for prediction.

   On the service details page, click the **Prediction** tab to use the service for prediction.

Figure 9-3 Prediction



9.2 Using a Large Model to Create a Model on ModelArts Standard and Deploy It as a Real-Time Service

Context

Currently, a large model can have hundreds of billions or even trillions of parameters, and its size becomes larger and larger. A large model with hundreds of billions of parameters exceeds 200 GB, and poses new requirements for version management and production deployment of the platform. For example, importing models requires dynamic adjustment of the tenant storage quota. Slow model loading and startup require a flexible timeout configuration in the deployment. The service recovery time needs to be shortened in the event that the model needs to be reloaded upon a restart caused by a load exception.

To address the preceding requirements, the ModelArts inference platform provides a solution to model management and service deployment in large model application scenarios.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Constraints

- You need to apply for the size quota of a model and add the whitelist cached using the local storage of the node.
- You need to use the custom engine **Custom** to configure dynamic loading.
- A dedicated resource pool is required to deploy the service.
- The disk space of the dedicated resource pool must be greater than 1 TB.

Procedure

1. **Applying for Increasing the Size Quota of a Model and Using the Local Storage of the Node to Cache the Whitelist**
2. **Uploading Model Data and Verifying the Consistency of Uploaded Objects**
3. **Creating a Dedicated Resource Pool**
4. **Creating a Model**
5. **Deploying a Real-Time Service**

Applying for Increasing the Size Quota of a Model and Using the Local Storage of the Node to Cache the Whitelist

During service deployment, the dynamically loaded model package is stored in the temporary disk space by default. When the service is stopped, the loaded files are deleted, and they need to be reloaded when the service is restarted. To avoid repeated loading, the platform allows the model package to be loaded from the local storage space of the node in the resource pool and keeps the loaded files valid even when the service is stopped or restarted (using the hash value to ensure data consistency).

To use a large model, you need to use a custom engine and enable dynamic loading when importing the model. In this regard, you need to perform the following operations:

- If the model size exceeds the default quota, submit a service ticket to increase the size quota of a single model. The default size quota of a model is 20 GB.
- Submit a service ticket to add the whitelist cached using the local storage of the node.

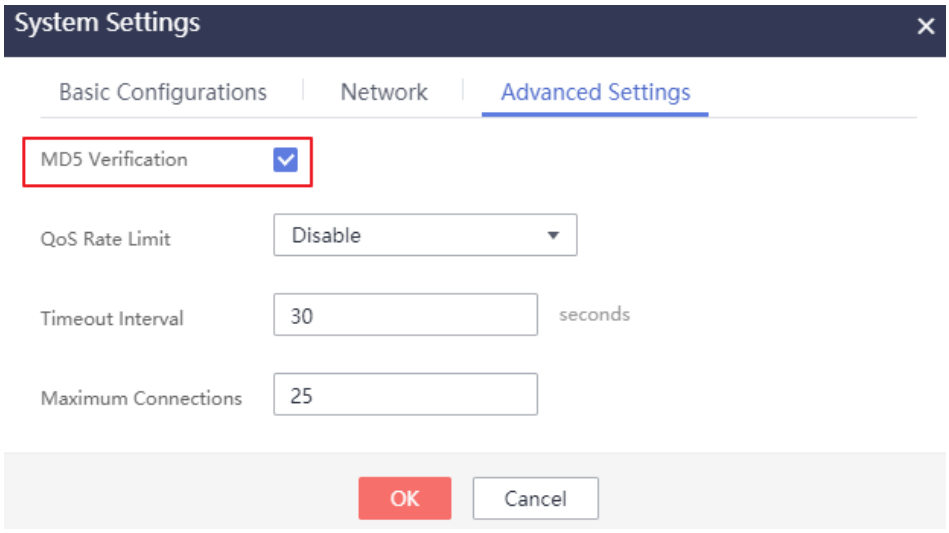
Uploading Model Data and Verifying the Consistency of Uploaded Objects

To ensure data integrity during dynamic loading, you need to verify the consistency of uploaded objects when uploading model data to OBS. obsutil, OBS Browser+, and OBS SDKs support verification of data consistency during upload. You can select a method that meets your requirements. For details, see **Verifying Data Consistency During Upload**.

For example, if you upload data via OBS Browser+, enable MD5 verification, as shown in **Figure 9-4**. When dynamic loading is enabled and the local persistent storage of the node is used, OBS Browser+ checks data consistency during data upload.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Figure 9-4 Configuring MD5 verification for OBS Browser+



Creating a Dedicated Resource Pool

To use the local persistent storage, you need to create a dedicated resource pool whose disk space is greater than 1 TB. You can view the disk information on the **Specifications** tab of the **Basic Information** page of the dedicated resource pool. If a service fails to be deployed and the system displays a message indicating that the disk space is insufficient, see **What Do I Do If Resources Are Insufficient When a Real-Time Service Is Deployed, Started, Upgraded, or Modified**.

Figure 9-5 Viewing the disk information of the dedicated resource pool



Creating a Model

If you use a large model to create a model and import the model from OBS, complete the following configurations:

1. Use a custom engine and enable dynamic loading.

   To use a large model, you need to use a custom engine and enable dynamic loading when importing the model. You can create a custom engine to meet special requirements for image dependency packages and inference frameworks in large model scenarios. For details about how to create a custom engine, see **Creating a Model Using a Custom Engine**.

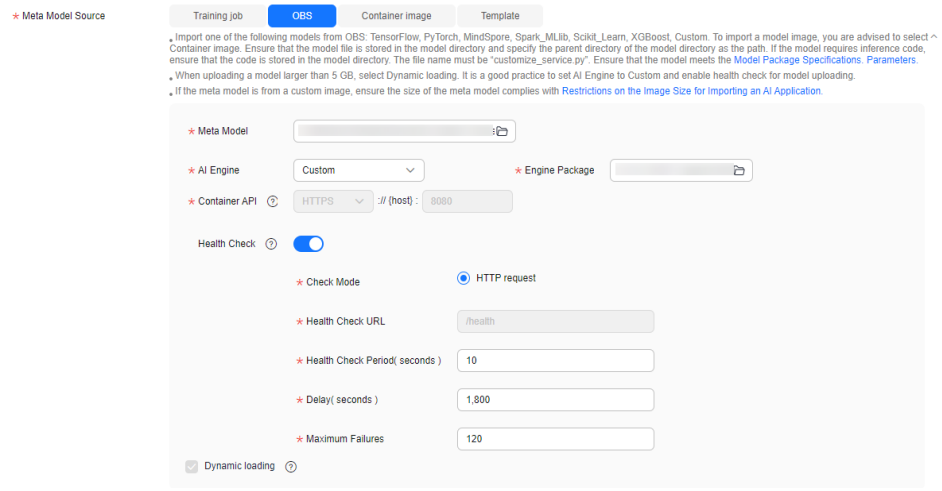
   When you use a custom engine, dynamic loading is enabled by default. The model package is separated from the image, and the model is dynamically loaded to the service load during service deployment.

2. Configure health check.

   Health check is mandatory for the models imported using a large model to identify unavailable services that are displayed as started.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Figure 9-6 Using a custom engine, enabling dynamic loading, and configuring health check



Deploying a Real-Time Service

When deploying the service, complete the following configurations:

1. Customize the deployment timeout interval.

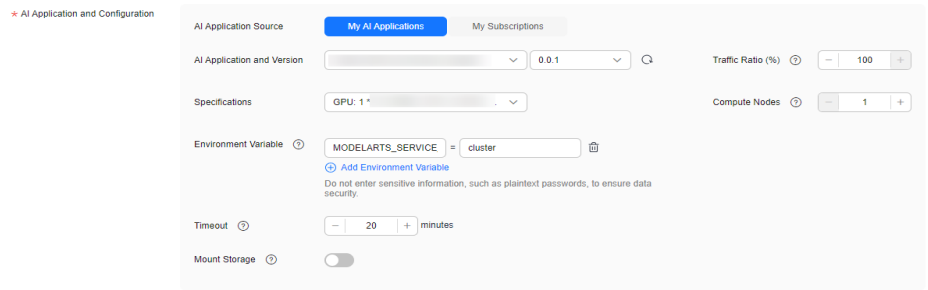
   Generally, the time for loading and starting a large model is longer than that for a common model. Set **Timeout** to a proper value. Otherwise, the timeout may elapse prior to the completion of the model startup, and the deployment may fail.

2. Add an environment variable.

   During service deployment, add the following environment variable to set the service traffic load balancing policy to cluster affinity, preventing unready service instances from affecting the prediction success rate:

   ```
   MODELARTS_SERVICE_TRAFFIC_POLICY: cluster
   ```

Figure 9-7 Customizing the deployment timeout interval and adding an environment variable



You are advised to deploy multiple instances to improve service reliability.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

9.3 Full-Process Development of WebSocket Real-Time Services

Context

WebSocket is a network transmission protocol that supports full-duplex communication over a single TCP connection. It is located at the application layer in an OSI model. The WebSocket communication protocol was established by IETF in 2011 as standard RFC 6455 and supplemented by RFC 7936. The WebSocket API in the Web IDL is standardized by W3C.

WebSocket simplifies data exchange between the client and the server and allows the server to proactively push data to the client. In the WebSocket API, if the initial handshake between the client and the server is successful, a persistent connection will be established between them and data can be transferred bidirectionally.

Prerequisites

- You are experienced in developing Java and familiar with JAR packaging.
- You have basic knowledge and calling methods of WebSocket.
- You are familiar with the method of creating an image using Docker.

Constraints

- WebSocket supports only the deployment of real-time services.
- It supports only real-time services deployed using models imported from custom images.

Preparations

Before using WebSocket in ModelArts for inference, bring your own custom image. The custom image must be able to provide complete WebSocket services in a standalone environment, for example, completing WebSocket handshakes and exchanging data between the client to the server. The model inference is implemented in the custom image, including downloading the model, loading the model, performing preprocessing, completing inference, and assembling the response body.

Procedure

To develop a WebSocket real-time service, perform the following operations:

- **Uploading the Image to SWR**
- **Creating a Model Using an Image**
- **Deploying the Model as a Real-Time Service**
- **Calling the WebSocket Real-Time Service**

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

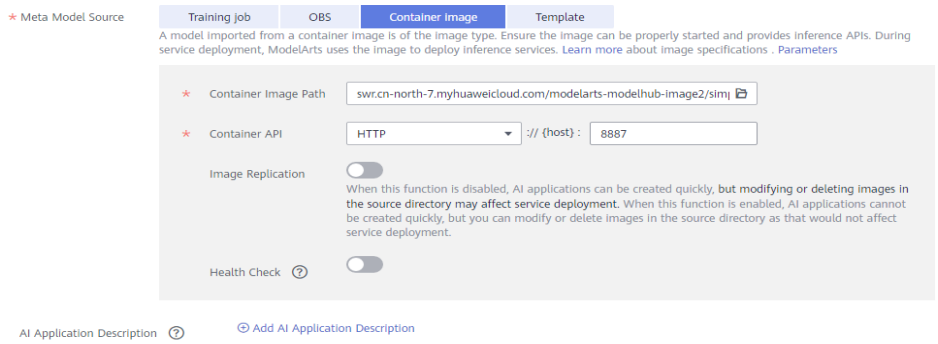
Uploading the Image to SWR

Upload the local image to SWR.

Creating a Model Using an Image

1. Log in to the ModelArts console. In the navigation pane on the left, choose **Model Management (AI Applications)**. On the displayed page, click **Create Model**.
2. Configure the AI application.
   - **Meta Model Source**: Select **Container image**.
   - **Container Image Path**: Select the path specified in **Uploading the Image to SWR**.
   - **Container API**: Configure this parameter based on site requirements.
   - **Health Check**: Retain default settings. If health check has been configured in the image, configure the health check parameters based on those configured in the image.

   Figure 9-8 Model configuration parameters



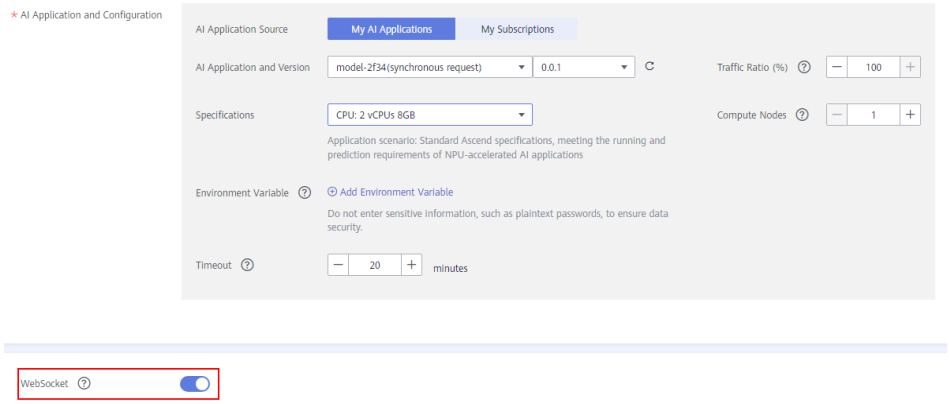
3. Click **Create now**. Wait until the model status changes to **Normal**.

Deploying the Model as a Real-Time Service

1. Log in to the ModelArts management console. In the navigation pane on the left, choose **Model Deployment** > **Real-Time Services**. On the displayed page, click **Deploy**.
2. Configure the service.
   - **Model and Version**: Select the model and version created in **Creating a Model Using an Image**.
   - **WebSocket**: Enable this function.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Figure 9-9 WebSocket



3. Click **Next**, confirm the configuration, and click **Submit**. In the real-time service list you will be redirected to, check the service status. When it changes to **Running**, the real-time service has been deployed.

Calling a WebSocket Real-Time Service

WebSocket itself does not require additional authentication. ModelArts WebSocket is WebSocket Secure-compliant, regardless of whether WebSocket or WebSocket Secure is enabled in the custom image. WebSocket Secure supports only one-way authentication, from the client to the server.

You can use one of the following authentication methods provided by ModelArts:

- **Token-based Authentication**
- **AK/SK**
- **App Authentication**

The following section uses GUI software Postman for prediction and token authentication as an example to describe how to call WebSocket.

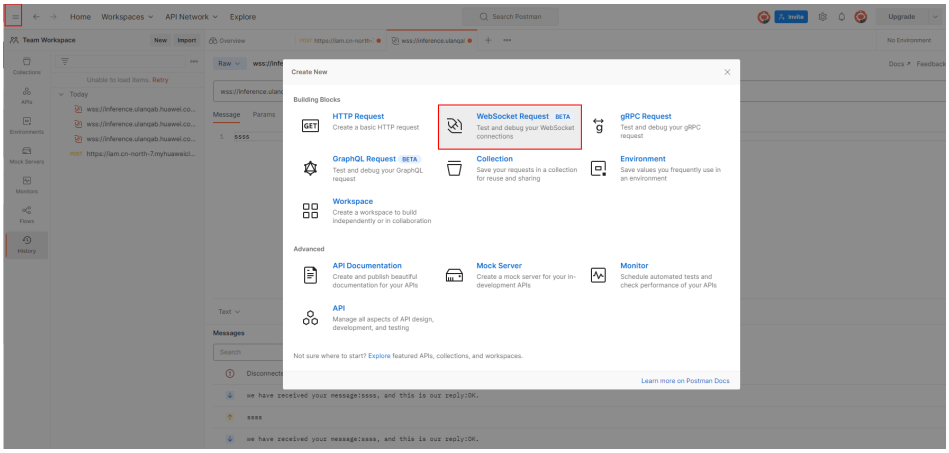
1. **Establish a WebSocket connection.**
2. **Exchange data between the WebSocket client and the server.**

Step 1 Establish a WebSocket connection.

1. Open Postman of a version later than 8.5, for example, 10.12.0. Click  in the upper left corner and choose **File** > **New**. In the displayed dialog box, select **WebSocket Request** (beta version currently).

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

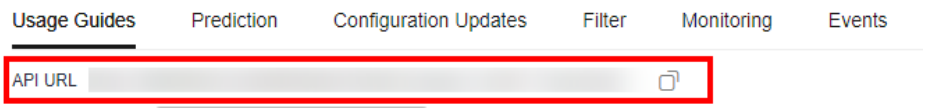
Figure 9-10 WebSocket Request



2. Configure parameters for the WebSocket connection.

   Select **Raw** in the upper left corner. Do not select **Socket.IO** (a type of WebSocket implementation, which requires that both the client and the server run on **Socket.IO**). In the address box, enter the **API Address** obtained on the **Usage Guides** tab on the service details page. If there is a finer-grained URL in the custom image, add the URL to the end of the address. If **queryString** is available, add this parameter in the **params** column. Add authentication information into the header. The header varies depending on the authentication mode, which is the same as that in the HTTPS-compliant inference service. Click **Connect** in the upper right corner to establish a WebSocket connection.

   Figure 9-11 Obtaining the API address

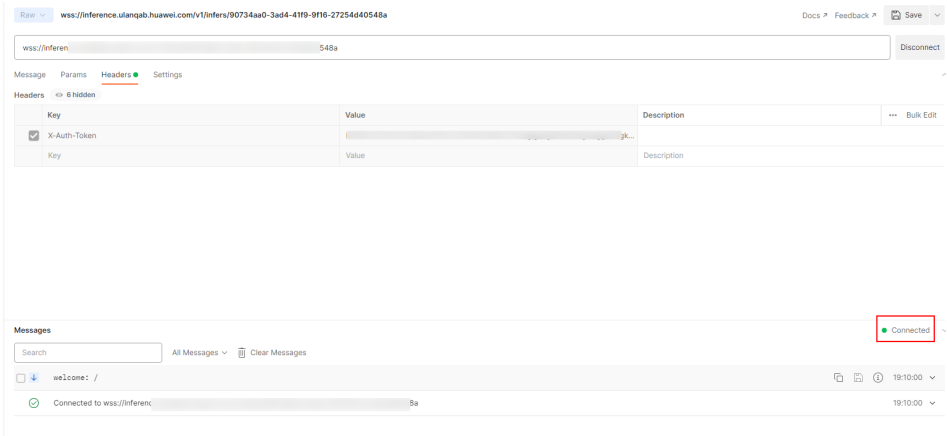
   NOTE

   - If the information is correct, **CONNECTED** will be displayed in the lower right corner.
   - If establishing the connection failed and the status code is 401, check the authentication.
   - If a keyword such as **WRONG_VERSION_NUMBER** is displayed, check whether the port configured in the custom image is the same as that configured in WebSocket or WebSocket Secure.

   The following shows an established WebSocket connection.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Figure 9-12 Connection established



NOTICE

Preferentially check the WebSocket service provided by the custom image. The type of implementing WebSocket varies depending on the tool you used. Possible issues are as follows: A WebSocket connection can be established but cannot be maintained, or the connection is interrupted after one request and needs to be reconnected. ModelArts only ensures that it will not affect the WebSocket status in a custom image (the API address and authentication mode may be changed on ModelArts).

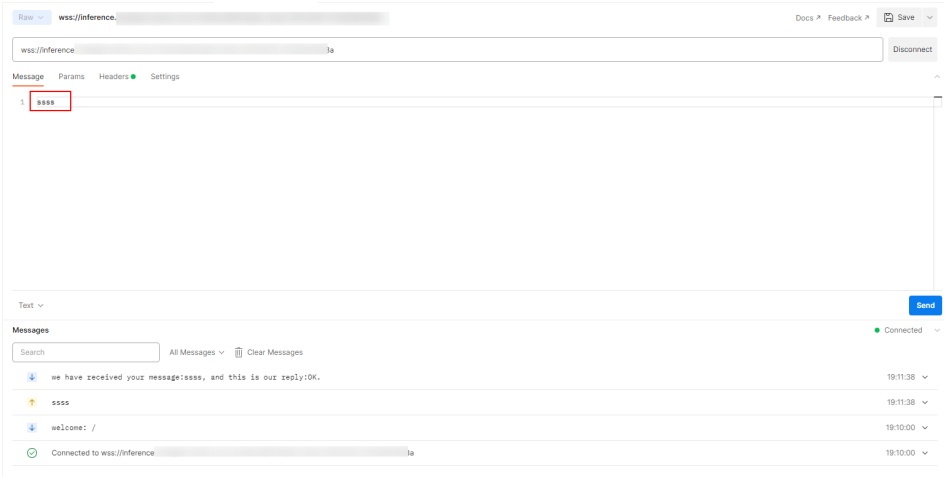
Step 2 Exchange data between the WebSocket client and the server.

After the connection is established, WebSocket uses TCP for full-duplex communication. The WebSocket client sends data to the server. The implementation types vary depending on the client, and the lib package may also be different for the same language. Different implementation types are not considered here.

The format of the data sent by the client is not limited by the protocol. Postman supports text, JSON, XML, HTML, and Binary data. Take text as an example. Enter the text data in the text box and click **Send** on the right to send the request to the server. If the text is oversized, Postman may be suspended.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Figure 9-13 Sending data



----End

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

10 Permissions Management

10.1 Basic Concepts

ModelArts allows you to configure fine-grained permissions for refined management of resources and permissions. This is commonly used by large enterprises, but it is complex for individual users. It is recommended that individual users configure permissions for using ModelArts by referring to **Configuring Agency Authorization for ModelArts with One Click**.

NOTE

If you meet any of the following conditions, read this document.

- You are an enterprise user with multiple departments. You need to specify users in different departments to access their dedicated resources. And there should be multiple roles, including administrator, algorithm developer, and application O&M engineer.
- You are an enterprise user and aim to restrict different roles to use specific functions. Logically, there should be multiple isolated environments, such as the development environment, pre-production environment, and production environment. You also want the operation permissions to vary depending on the environment. Or you need to control permissions of specific IAM user or user group.
- You are an individual user, and you have created multiple IAM users. You need to assign different ModelArts permissions to different IAM users.
- You need to understand the concepts and operations of ModelArts permissions management.

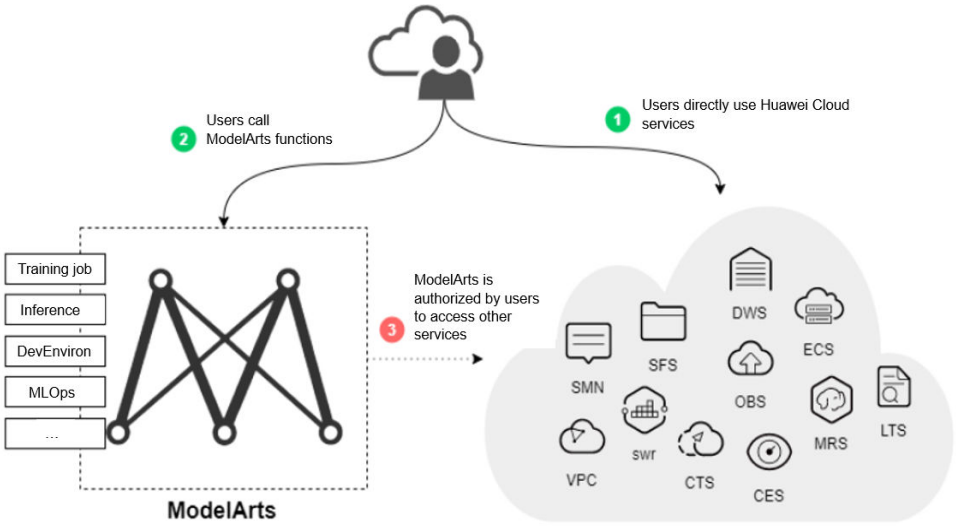
ModelArts uses Identity and Access Management (IAM) for most permissions management functions. Before reading below, learn about **Basic Concepts**. This helps you better understand this document.

To implement fine-grained permissions management, ModelArts provides permission control, agency authorization, and workspace. The following describes the details.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

ModelArts Permissions and Agencies

Figure 10-1 Permissions management



Like other services, ModelArts functions are controlled by IAM permissions. For example, if you as an IAM user need to create a training job on ModelArts, you must have the **modelarts:trainJob:create** permission. For details about how to assign permissions to a user (you need to add the user to a user group and then assign permissions to the user group), see **Permissions Management**.

ModelArts must access other services for AI computing. For example, ModelArts must access OBS to read your data for training. For security purposes, ModelArts must be authorized to access other cloud services. This is agency authorization.

The following summarizes permissions management:

- Your access to any cloud service is controlled through IAM. You must have the permissions of the cloud service. (The required service permissions vary depending on the functions you use.)
- To use ModelArts functions, you need to grant permissions through IAM.
- ModelArts must be authorized by you to access other cloud services for AI computing.

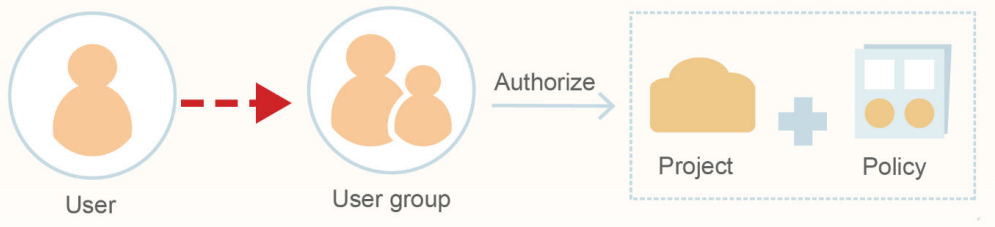
ModelArts Permissions Management

By default, new IAM users do not have any permissions assigned. You need to add the user to a user group and grant the user group with policies, so that the users in the group can inherit the permissions. After authorization, users can perform operations on ModelArts based on permissions.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

CAUTION

- ModelArts is a project-level service deployed and accessed in specific physical regions. When you authorize an agency, you can set the scope for the permissions you select to all resources, enterprises projects, or region-specific projects. If you specify region-specific projects, the selected permissions will be applied to resources in these projects.
- ModelArts supports enterprise projects. You can specify an enterprise project when selecting the authorization scope. For details, see **Creating a User Group and Assigning Permissions**.



When assigning permissions to a user group, IAM does not directly assign specific permissions to the user group. Instead, IAM needs to add the permissions to a policy and then assign the policy to the user group. To facilitate user permissions management, each cloud service provides some preset policies for you to directly use. If the preset policies cannot meet your requirements of fine-grained permissions management, you can customize policies.

Table 10-1 lists all the preset system-defined policies supported by ModelArts.

Table 10-1 System-defined policies supported by ModelArts

Policy	Description	Type
ModelArts FullAccess	Administrator permissions for ModelArts. Users granted these permissions can operate and use ModelArts.	System-defined policy
ModelArts CommonOperations	Common user permissions for ModelArts. Users granted these permissions can operate and use ModelArts, but cannot manage dedicated resource pools.	System-defined policy
ModelArts Dependency Access	Permissions on dependent services for ModelArts	System-defined policy

Generally, ModelArts FullAccess is assigned only to administrators. If fine-grained management is not required, assigning ModelArts CommonOperations to all users will meet the development requirements of most small teams. If you want to customize policies for fine-grained permissions management, see **IAM**.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

NOTE

When you assign ModelArts permissions to a user, the system does not automatically assign the permissions of other services to the user. This ensures security and prevents unexpected unauthorized operations. In this case, however, you must separately assign permissions of different services to users so that they can perform some ModelArts operations.

For example, if an IAM user needs to use OBS data for training and the ModelArts training permission has been configured for the IAM user, the IAM user still needs to be assigned with the OBS read, write, and list permissions. The OBS list permission allows you to select the training data path on ModelArts. The read permission is used to preview data and read data for training. The write permission is used to save training results and logs.

- For individual users or small organizations, it is a good practice to configure the **Tenant Administrator** policy that applies to global services for IAM users. In this way, IAM users can obtain all service permissions except IAM. However, this may cause security issues. (For an individual user, its default IAM user belongs to the **admin** user group and has the **Tenant Administrator** permission.)
- If you want to restrict user operations, configure the minimum permissions of OBS for ModelArts users. For details, see **OBS Permissions Management**. For details about fine-grained permissions management of other cloud services, see the corresponding cloud service documents.

Because AI technology evolves quickly, ModelArts continuously releases new features. Preset policies may not immediately include all new permissions. If you encounter an insufficient permissions error, go to the IAM console and manually add the required permissions.

ModelArts Agency Authorization

ModelArts must be authorized by users to access other cloud services for AI computing. In the IAM permission system, such authorization is performed through agencies.

For details about the basic concepts and operations of agencies, see **Cloud Service Delegation**.

To simplify agency authorization, ModelArts supports automatic agency authorization configuration. You only need to configure an agency for yourself or specified users on the **Permission Management** page of the **ModelArts console**.

NOTE

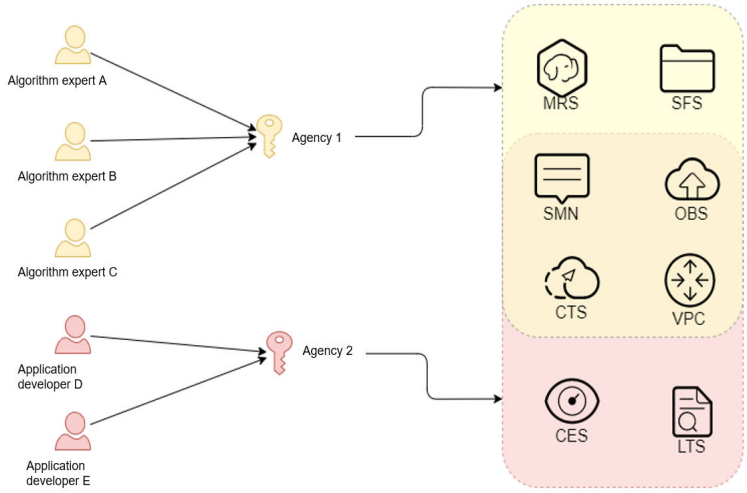
- Only users with the IAM agency management permission can perform this operation. Generally, members in the IAM admin user group have this permission.
- ModelArts agency authorization is region-specific, which means that you must perform agency authorization in each region you use.

On the **Permission Management** page of the **ModelArts console**, after you click **Add Agency**, you can configure an agency for a specific user or all users. Generally, an agency named **modelarts_agency**_*<username>*_*<random-ID>* is created by default. In the **Permissions** area, you can select the preset permission configuration or select the required policies. If both options cannot meet your requirements, you can create an agency on the IAM management page (you need to delegate ModelArts to access your resources), and then use an existing agency instead of adding an agency on the **Add Agency** page.

ModelArts associates multiple users with one agency. This means that if two users need to configure the same agency, you do not need to create an agency for each user. Instead, you only need to configure the same agency for the two users.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Figure 10-2 Mapping between users and agencies



NOTE

Each user can use ModelArts only after being associated with an agency. However, even if the permissions assigned to the agency are insufficient, no error is reported when the API is called. An error occurs only when the system uses unauthorized functions. For example, you enable message notification when creating a training job. Message notification requires SMN authorization. However, an error occurs only when messages need to be sent for the training job. The system ignores some errors, and other errors may cause job failures. When you implement permission minimization, ensure that you will still have sufficient permissions for the required operations on ModelArts.

Strict Authorization

In strict authorization mode, explicit authorization by the account administrator is required for IAM users to access ModelArts. The administrator can add the required ModelArts permissions to common users through authorization policies.

In non-strict authorization mode, IAM users can use ModelArts without explicit authorization. The administrator needs to configure the deny policy for IAM users to prevent them from using some ModelArts functions.

In this mode, IAM users must be authorized to use ModelArts functions. In this way, the permission scope of IAM users can be accurately controlled, minimizing permissions granted to IAM users. The strict authorization mode is recommended.

By default, a newly created account uses the strict authorization mode. To change the strict authorization mode, go to the permissions management page of the old ModelArts console. The new console does not provide the strict authorization switch.

Managing Resource Access Using Workspaces

Workspace enables enterprise customers to split their resources into multiple spaces that are logically isolated and to manage access to different spaces.

After workspace is enabled, a default workspace is created. All resources you have created are in this workspace. A workspace is like a ModelArts twin. You can

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

switch between workspaces in the upper left corner of the ModelArts console. Jobs in different workspaces do not affect each other.

When creating a workspace, you must bind it to an enterprise project. Multiple workspaces can be bound to the same enterprise project, but one workspace cannot be bound to multiple enterprise projects. You can use workspaces for refined restrictions on resource access and permissions of different users. The restrictions are as follows:

- Users must be authorized to access specific workspaces (this must be configured on the pages for creating and managing workspaces). This means that access to AI assets such as datasets and algorithms can be managed using workspaces.
- In the preceding permission authorization operations, if you set the scope to enterprise projects, the authorization takes effect only for workspaces bound to the selected projects.

NOTE

- Restrictions on workspaces and permission authorization take effect at the same time. That is, a user must have both the permission to access the workspace and the permission to create training jobs (the permission applies to this workspace) so that the user can submit training jobs in this workspace.
- If you have enabled an enterprise project but have not enabled a workspace, all operations are performed in the default enterprise project. Ensure that the permissions on the required operations apply to the default enterprise project.
- The preceding restrictions do not apply to users who have not enabled any enterprise project.

Summary

Key features of ModelArts permissions management:

- If you are an individual user, you do not need to consider fine-grained permissions management. Your account has all permissions to use ModelArts by default.
- All functions of ModelArts are controlled by IAM. You can use IAM authorization to implement fine-grained permissions management for specific users.
- All users (including individual users) can use specific functions only after agency authorization on ModelArts (**Settings** > **Add Agency**). Otherwise, unexpected errors may occur.
- If you have enabled the enterprise project function, you can also enable ModelArts workspace and use both basic authorization and workspace for refined permissions management.

10.2 Permission Management Mechanisms

10.2.1 IAM

This section describes the IAM permission configurations for all ModelArts functions.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

IAM Permissions

If you need to assign different permissions to employees in your enterprise to access your purchased ModelArts resources, Identity and Access Management (IAM) is a good choice for fine-grained permissions management. IAM provides identity authentication, fine-grained permissions management, and access control. IAM helps you secure access to your cloud resources. If your Huawei account can meet your requirements and you do not need an IAM account to manage user permissions, skip this chapter.

IAM is a free service. You only pay for the resources in your account.

With IAM, you can assign permissions to control users' access to specific resources. For example, if the software developers in your enterprise need to own permissions to use ModelArts, yet you do not want them to own high-risk operation permissions such as deleting ModelArts, you can grant permissions using IAM to limit their permission on ModelArts.

For details about IAM, see **What is IAM?**

Role/Policy-based Authorization

ModelArts supports role/policy-based authorization. By default, new IAM users do not have any permissions. You need to add a user to one or more groups, and assign permissions policies or roles to these groups. Users inherit permissions of the groups to which they are added. This process is called authorization. The users then inherit permissions from the groups and can perform specified operations on cloud services.

ModelArts is a project-level service deployed for specific regions. When you set **Scope** to **Region-specific projects** and select the specified projects (for example, **ap-southeast-2**) in the specified regions (for example, **AP-Bangkok**), the users only have permissions for ModelArts resources in the selected projects. If you set **Scope** to **All resources**, the users have permissions for ModelArts resources in all region-specific projects. When accessing ModelArts, the users need to switch to a region where they have been authorized to use cloud services.

Table 10-2 lists all system-defined policies supported by ModelArts. If preset ModelArts permissions cannot meet your requirements, create a custom policy by referring to **Policy Fields in JSON Format**.

Table 10-2 System-defined policies supported by ModelArts

Policy	Description	Type
ModelArts FullAccess	All permissions for ModelArts administrators	System-defined policy
ModelArts CommonOperations	All operation permissions for ModelArts common users, which do not include managing dedicated resource pools.	System-defined policy
ModelArts Dependency Access	Permissions on dependency services for ModelArts	System-defined policy

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

ModelArts depends on other cloud services. To check or view the cloud services, configure the corresponding permissions on the ModelArts console. The dependencies are as follows.

Table 10-3 ModelArts dependencies

Console Function	Dependency
Data Preparation	MapReduce Service (MRS) Object Storage Service (OBS) SoftWare Repository for Container (SWR)
Data Management	MapReduce Service (MRS) Data Warehouse Service (DWS) Data Lake Insight (DLI) Object Storage Service (OBS) Data Warehouse Service (DWS) ModelArts
Development Environment	NAT Gateway Object Storage Service (OBS) Cloud Trace Service (CTS) Elastic Cloud Server (ECS) SoftWare Repository for Container (SWR) Scalable File Service (SFS) Application Operations Management (AOM) Data Encryption Workshop (DEW) Bare Metal Server (BMS) Image Management Service (IMS) ModelArts Virtual Private Cloud (VPC) Elastic Volume Service (EVS) CloudMatrix
Training	Object Storage Service (OBS) Simple Message Notification (SMN) Cloud Trace Service (CTS) SoftWare Repository for Container (SWR) Application Operations Management (AOM) Key Management Service (KMS) Virtual Private Cloud (VPC) ModelArts Dedicated Distributed Storage Service (DSS) SFS Turbo

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Console Function	Dependency
Inference (Old Version)	Application Operations Management (AOM) Data Lake Insight (DLI) Simple Message Notification (SMN) Object Storage Service (OBS) Intelligent EdgeFabric (IEF) Log Tank Service (LTS) Cloud Trace Service (CTS) SoftWare Repository for Container (SWR) VPC Endpoint (VPCEP)
Model Deployment (Old Version)	Data Encryption Workshop (DEW) ModelArts Object Storage Service (OBS) Virtual Private Cloud (VPC) Log Tank Service (LTS) SoftWare Repository for Container (SWR) Elastic Load Balance (ELB) SFS Turbo VPC Endpoint (VPCEP)
Real-Time Service (New Version)	Virtual Private Cloud (VPC) VPC Endpoint (VPCEP) Object Storage Service (OBS) SoftWare Repository for Container (SWR) Scalable File Service Turbo (SFS Turbo) Key Management System (KMS) Elastic Load Balance (ELB) Cloud Eye Simple Message Notification (SMN) Enterprise Project Management Service (EPS) Cloud Trace Service (CTS) Log Tank Service (LTS) Application Operations Management (AOM) ModelArts
AI Gallery	Object Storage Service (OBS) Cloud Trace Service (CTS)

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Console Function	Dependency
Dedicated Resource Pool	Object Storage Service (OBS) Dedicated Distributed Storage Service (DSS) Virtual Private Cloud (VPC) SoftWare Repository for Container (SWR)
Lite Cluster	Elastic Cloud Server (ECS) Application Performance Management (APM) Elastic Volume Service (EVS) Data Encryption Workshop (DEW) Application Operations Management (AOM) Bare Metal Server (BMS) Image Management Service (IMS) Cloud Container Engine (CCE) Dedicated Distributed Storage Service (DSS) Virtual Private Cloud (VPC) Log Tank Service (LTS) SoftWare Repository for Container (SWR) SFS Turbo CloudMatrix

If system-defined policies cannot meet your requirements, you can create a custom policy. For details about the actions supported by custom policies, see **ModelArts Resource Permissions**.

To create a custom policy, choose either visual editor or JSON.

- Visual editor: Select cloud services, actions, resources, and request conditions without the need to know policy syntax.
- JSON: Create a JSON policy or edit an existing one.

For details, see **Creating a Custom Policy**. This section provides examples of common custom ModelArts policies.

- Example 1: Grant permission to manage images.

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "modelarts:image:register",
                "modelarts:image:listGroup"
            ]
        }
    ]
}
```

- Example 2: Grant permission to deny creating, updating, and deleting a dedicated resource pool.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

A policy with only "Deny" permissions must be used together with other policies. If the permissions granted to an IAM user contain both "Allow" and "Deny", the "Deny" permissions take precedence over the "Allow" permissions.

```
{
   "Version": "1.1",
   "Statement": [
      {
         "Action": [
            "modelarts:*:*"
         ],
         "Effect": "Allow"
      },
      {
         "Action": [
            "swr:*:*"
         ],
         "Effect": "Allow"
      },
      {
         "Action": [
            "smn:*:*"
         ],
         "Effect": "Allow"
      },
      {
         "Action": [
            "modelarts:pool:create",
            "modelarts:pool:update",
            "modelarts:pool:delete"
         ],
         "Effect": "Deny"
      }
   ]
}
```

- Example 3: Create a custom policy containing multiple actions.

  A custom policy can contain actions of multiple services that are of the global or project-level type. The following is an example policy containing actions of multiple services:

```
{
   "Version": "1.1",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "modelarts:service:*"
         ]
      },
      {
         "Effect": "Allow",
         "Action": [
            "lts:logs:list"
         ]
      }
   ]
}
```

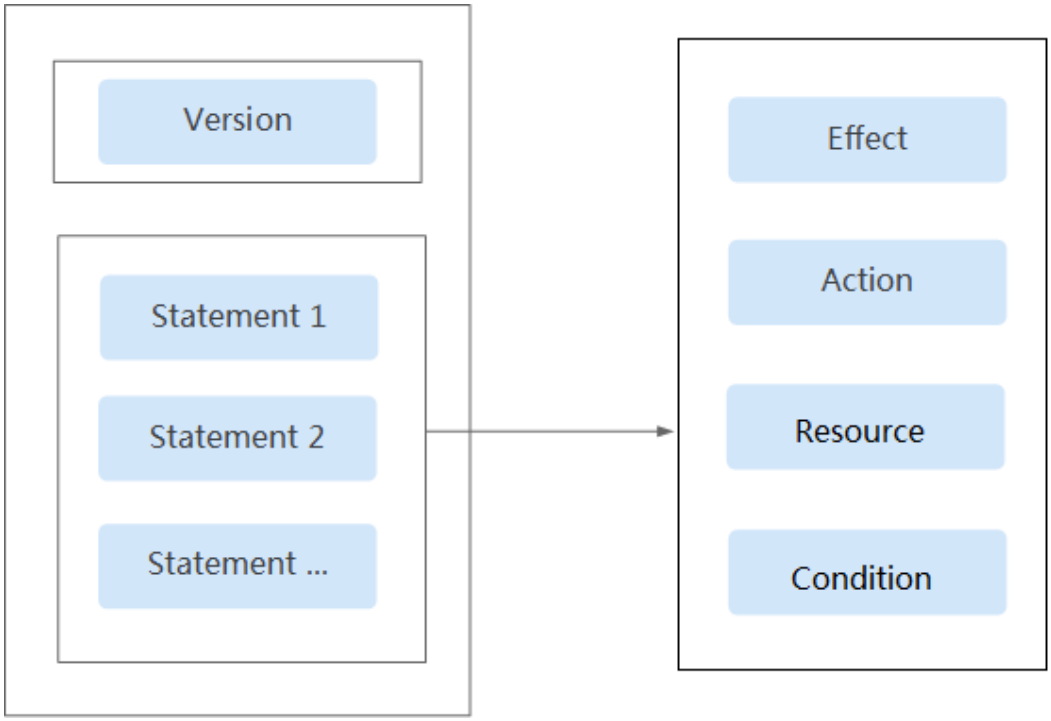
Policy Fields in JSON Format

Policy Structure

A policy consists of a version and one or more statements (indicating different actions).

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Figure 10-3 Policy structure



Policy Parameters

The following describes policy parameters. You can create custom policies by specifying the parameters. For details, see **Custom Policy Use Cases**.

Table 10-4 Policy parameters

Parameter		Description	Value
Version		Policy version	**1.1**: indicates policy-based access control.
Statement: authorization statement of a policy	Effect	Whether to allow or deny the operations defined in the action	• **Allow**: indicates the operation is allowed. • **Deny**: indicates the operation is not allowed. **NOTE** If the policy used to grant user permissions contains both **Allow** and **Deny** for the same action, **Deny** takes precedence.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Parameter		Description	Value
	Action	Operation to be performed on the service	Format: "*Service name*:*Resource type*:*Action*". Wildcard characters (*) are supported, indicating all options. Example: **modelarts:notebook:list** indicates the permission to view a notebook instance list. **modelarts** indicates the service name, **notebook** indicates the resource type, and **list** indicates the operation. View all actions of a service in its *API Reference*.
	Condition	Condition for a policy to take effect, including **condition keys and operators**	Format: "*Condition operator*:{*Condition key*:[*Value 1*,*Value 2*]}" If you set multiple conditions, the policy takes effect only when all the conditions are met. Example: **StringEndWithIfExists": {"g:UserName":["specialCharacter"]}**: The statement is valid for users whose names end with **specialCharacter**.
	Resource	Resources on which a policy takes effect	Format: *Service name*:<*Region*>:<*Account ID*>:*Resource type*:*Resource path*. Asterisks (*) are supported for resource type, indicating all resources. **NOTE** ModelArts authorization does not allow you to specify a resource path.

ModelArts Resource Types

Administrators can specify the scope based on ModelArts resource types. The following table lists the resource types supported by ModelArts:

Table 10-5 Resource types supported by ModelArts role/policy-based authorization

Resource Type	Description
notebook	Notebook instances in DevEnviron
exemlProject	ExeML projects

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Resource Type	Description
exemlProjectInf	ExeML-powered real-time inference service
exemlProjectTrain	ExeML-powered training jobs
exemlProjectVersion	ExeML project version
pool	Dedicated Resource Pool
network	Networking of a dedicated resource pool
trainJob	Training job
trainJobLog	Runtime logs of a training job
trainJobInnerModel	Preset model
model	Models
service	Real-time service
nodeservice	Edge service
workspace	Workspace
dataset	Dataset
dataAnnotation	Dataset labels
aiAlgorithm	Algorithm for training jobs
image	Image
devserver	Elastic BMS

ModelArts Resource Permissions

For details, see "Permissions Policies and Supported Actions" in *ModelArts API Reference*.

- **Data Management Permissions**
- **DevEnviron Permissions**
- **Training Job Permissions**
- **Model Management Permissions**
- **Service Management Permissions**

10.2.2 Dependencies and Agencies

Function Dependency

Function Dependency Policies

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

When using ModelArts, you may be required to use other cloud services. For example, before submitting a training job, you must select OBS paths for storing the dataset and logs, respectively. Therefore, when configuring fine-grained authorization policies for a user, the administrator must configure dependent permissions so that the user can use required functions.

NOTE

- If you use ModelArts as the root user (default IAM user with the same name as the account), the root user has all permissions by default.
- Ensure that the current user has the dependency policy permissions for agency authorization. For example, if you want to grant the SWR Admin permission to a ModelArts agency, ensure that you have the SWR Admin permission.

Table 10-6 Basic configuration

Application Scenario	Dependency Service	Dependency Policy	Supported Function
Global configuration	IAM	iam:users:listUsers	Obtain a user list. This action is required by the administrator only.
Basic function	IAM	iam:tokens:assume	(Mandatory) Use an agency to obtain temporary authentication credentials.
Basic function	BSS	bss:balance:view	Show the balance of the current account on the page after resources are created on the ModelArts console.

Table 10-7 Managing workspaces

Application Scenario	Dependency Service	Dependency Policy	Supported Function
Workspace	IAM	iam:users:listUsers	Authorize an IAM user to use a workspace.
	ModelArts	modelarts:*:delete*	Clear resources in a workspace when deleting it.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Table 10-8 Managing notebook instances

Application Scenario	Dependency Service	Dependency Policy	Supported Function
Lifecycle management of development environment instances	ModelArts	modelarts:notebook:create modelarts:notebook:list modelarts:notebook:get modelarts:notebook:update modelarts:notebook:delete modelarts:notebook:start modelarts:notebook:stop modelarts:notebook:updateStopPolicy modelarts:image:delete modelarts:image:list modelarts:image:create modelarts:image:get modelarts:pool:list modelarts:tag:list modelarts:network:get	Start, stop, create, delete, and update an instance. After the **modelarts:notebook:list** or **modelarts:notebook:get** permission is configured, you can open a notebook instance by default.
	AOM	aom:metric:get aom:metric:list aom:alarm:list	
	VPC	vpc:securityGroups:get vpc:vpcs:list vpc:securityGroups:get vpc:vpcs:list	

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Application Scenario	Dependency Service	Dependency Policy	Supported Function
Dynamically mounting storage	ModelArts	modelarts:notebook:listMountedStorages modelarts:notebook:mountStorage modelarts:notebook:getMountedStorage modelarts:notebook:umountStorage	Dynamically mount storage.
	OBS	obs:bucket:ListAllMyBuckets obs:bucket:ListBucket	
Image management	ModelArts	modelarts:image:register modelarts:image:listGroup	Register and view an image on the **Image Management** page.
Saving an image	SWR	SWR Admin	The **SWR Admin** policy contains the maximum scope of SWR permissions, which can be used to: • Save a running development environment instance as an image. • Create a notebook instance using a custom image.
Using the SSH function	ECS	ecs:serverKeypairs:list ecs:serverKeypairs:get ecs:serverKeypairs:delete ecs:serverKeypairs:create	Configure a login key for a notebook instance.
	DEW	kps:domainKeypairs:get kps:domainKeypairs:list kps:domainKeypairs:createkmskey	
	KMS	kms:cmk:list	

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Application Scenario	Dependency Service	Dependency Policy	Supported Function
Mounting an SFS Turbo file system	SFS Turbo	SFS Turbo FullAccess	Read and write an SFS directory as an IAM user. Mount an SFS file system that is not created by you to a notebook instance using a dedicated resource pool.
Viewing all Instances	ModelArts	modelarts:notebook:listAllNotebooks	View development environment instances of all users on the ModelArts management console. This action is required by the development environment instance administrator.
	IAM	iam:users:listUsers	
Local VS Code plug-in or PyCharm Toolkit	ModelArts	modelarts:notebook:listAllNotebooks modelarts:trainJob:create modelarts:trainJob:list modelarts:trainJob:update modelarts:trainJobVersion:delete modelarts:trainJob:get modelarts:trainJob:logExport modelarts:workspace:getQuotas (This policy is required if the **workspace** function is enabled.)	Access a notebook instance from local VS Code and submit training jobs.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Application Scenario	Dependency Service	Dependency Policy	Supported Function
	OBS	obs:bucket:ListAllMybuckets obs:bucket:HeadBucket obs:bucket:ListBucket obs:bucket:GetBucketLocation obs:object:GetObject obs:object:GetObjectVersion obs:object:PutObject obs:object:DeleteObject obs:object:DeleteObjectVersion obs:object:ListMultipartUploadParts obs:object:AbortMultipartUpload obs:object:GetObjectAcl obs:object:GetObjectVersionAcl obs:bucket:PutBucketAcl obs:object:PutObjectAcl obs:object:ModifyObjectMetaData	
	IAM	iam:projects:listProjects	Obtain an IAM project list through local PyCharm for access configurations.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Table 10-9 Lite Server

Application Scenario	Dependency Service	Dependency Policy	Supported Function
Managing the Lifecycle of Lite Server instances	ModelArts	modelarts:devserver:create modelarts:devserver:listByUser modelarts:devserver:list modelarts:devserver:get modelarts:devserver:delete modelarts:devserver:start modelarts:devserver:stop modelarts:devserver:sync	Create, start, and stop an instance, obtain the instance list, obtain all instances of a tenant, obtain instance details, and synchronize instance status.
	ECS	ecs:serverKeypairs:createecs:*:get ecs:cloudServers:redeploy	
	BMS	bms:servers:redeploy	
	IAM	iam:users:getUser iam:users:listUsers iam:projects:listProjects	
	VPC	vpc.*.list	
	EPS	eps.*.list	
	EVS	evs.*.list	
	IMS	ims.*.list ims.*.get	

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Application Scenario	Dependency Service	Dependency Policy	Supported Function
	CloudMatrix	cloudmatrix:hyperinstance:create cloudmatrix:hyperinstance:delete cloudmatrix:hyperinstance:get cloudmatrix:hyperinstance:list cloudmatrix:hyperinstance:attachNodeVolume cloudmatrix:hyperinstance:detachNodeVolume cloudmatrix:hyperinstance:changeOs cloudmatrix:hyperinstance:start cloudmatrix:hyperinstance:stop cloudmatrix:hyperinstance:liveScaleUp cloudmatrix:hyperinstance:liveScaleDown cloudmatrix:operation:get cloudmatrix:hyperinstance:startNode cloudmatrix:hyperinstance:stopNode cloudmatrix:hyperinstance:rebootNode cloudmatrix:hyperinstance:updateNode cloudmatrix:hyperinstance:changeNodeOs cloudmatrix:hyperinstance:reinstallNodeOs cloudmatrix:hyperinstanceCluster:create cloudmatrix:hyperinstanceCluster:list	Must be configured when a ModelArts Lite Server uses CloudMatrix supernodes. These permissions are required for managing the lifecycle of CloudMatrix supernodes.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Application Scenario	Dependency Service	Dependency Policy	Supported Function
		cloudmatrix:hyperinstanceCluster:get cloudmatrix:hyperinstance:batchCreateTags cloudmatrix:hyperinstance:batchDeleteTags cloudmatrix:hyperinstance:listTags cloudmatrix:scheduledEvent:create cloudmatrix:scheduledEvent:list cloudmatrix:scheduledEvent:accept cloudmatrix:scheduledEvent:cancel cloudmatrix:hyperinstance:bindNodePublicIp	

Table 10-10 Managing training jobs

Application Scenario	Dependency Service	Dependency Policy	Supported Function
Training management	ModelArts	modelarts:brain:showLogDiagnosisJob modelarts:brain:dropLogDiagnosisJob modelarts:brain:updateLogDiagnosisJob modelarts:brain:stopLogDiagnosisJob modelarts:brain:createLogDiagnosisJob modelarts:brain:listLogDiagnosisJobs modelarts:brain:createDiagnosisTask modelarts:brain:showDiagnosisTask	Create, query, delete, terminate, and modify training log diagnosis.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Application Scenario	Dependency Service	Dependency Policy	Supported Function
		modelarts:workspace:getQuota modelarts:workspace:get	Obtain a workspace quota. This policy is required if **workspace** function is enabled.
	VPC	vpc:addressGroups:get vpc:firewalls:get vpc:flowLogs:get vpc:peerings:get vpc:privateIps:get vpc:subNetworkInterfaces:get vpc:subnets:getReservation vpc:addressGroups:list vpc:firewalls:list vpc:privateIps:list vpc:quotas:list vpc:subNetworkInterfaces:list vpc:subnets:list vpc:ports:get vpc:routeTables:get vpc:networks:get vpc:securityGroupRules:get vpc:securityGroups:get vpc:subnets:get vpc:vpcs:get vpc:routes:get vpc:routes:list vpc:routeTables:list vpc:vpcs:list	Create and delete networks, and interconnect VPCs.
	DSS	dss:pools:get dss:pools:list	Query the list and details of DSS storage pools.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Application Scenario	Dependency Service	Dependency Policy	Supported Function
	SFS Turbo	sfsturbo:shares:getShare sfsturbo:shares:getAllShares sfsturbo:shares:getTag sfsturbo:shares:getAllTag sfsturbo:shares:getDataRepositoryTask sfsturbo:shares:getAllDataRepositoryTasks sfsturbo:shares:getJob sfsturbo:shares:getFlavors sfsturbo:shares:getQuota sfsturbo:shares:getAZInfo	Use SFS Turbo in a training job.
	SWR	swr:repo:getRepo swr:repo:listRepoTags swr:repo:upload swr:repo:download swr:system:createLoginSecret swr:instance:createTempCredential swr:namespace:getNamespace swr:repository:downloadArtifact	Use a custom image to create a training job.
	SMN	smn:topic:publish smn:topic:list smn:template:list smn:template:create	Notify training job status changes through SMN.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Application Scenario	Dependency Service	Dependency Policy	Supported Function
	OBS	obs:bucket:ListAllMybuckets obs:bucket:HeadBucket obs:bucket:ListBucket obs:bucket:GetBucketLocation obs:bucket:ListBucketVersions obs:object:GetObject obs:object:GetObjectVersion obs:object:PutObject obs:object:DeleteObject obs:object:DeleteObjectVersion obs:object:ListMultipartUpload-Parts obs:object:AbortMultipartUp-load obs:object:GetObjectAcl obs:object:GetObjectVersionAcl obs:bucket:PutBucketAcl obs:object:PutObjectAcl	Run a training job using a dataset in an OBS bucket.

Table 10-11 Model management

Application Scenario	Dependency Service	Dependency Policy	Supported Function
Model management	SWR	SWR Admin	Use a custom engine when you import a model from a custom image or OBS. SWR shared edition does not support fine-grained permissions. Therefore, the administrator permission is required.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Application Scenario	Dependency Service	Dependency Policy	Supported Function
	OBS	obs:bucket:ListAllMybuckets (Obtaining a bucket list) obs:bucket:HeadBucket (Obtaining bucket metadata) obs:bucket:ListBucket (Listing objects in a bucket) obs:bucket:GetBucketLocation (Obtaining the bucket location) obs:object:GetObject (Obtaining object content and metadata) obs:object:GetObjectVersion (Obtaining object content and metadata) obs:object:PutObject (Uploading objects using PUT method, uploading objects using POST method, copying objects, appending an object, initializing a multipart task, uploading parts, and merging parts) obs:object:DeleteObject (Deleting an object or batch deleting objects) obs:object:DeleteObjectVersion (Deleting an object or batch deleting objects) obs:object:ListMultipartUpload-Parts (Listing uploaded parts) obs:object:AbortMultipartUpload (Aborting multipart uploads) obs:object:GetObjectAcl (Obtaining an object ACL) obs:object:GetObjectVersionAcl (Obtaining an object ACL) obs:bucket:PutBucketAcl (Configuring a bucket ACL) obs:object:PutObjectAcl (Configuring an object ACL)	Import a model from OBS. Specify an OBS path for model conversion.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Table 10-12 Inference deployment (Old Version)

Application Scenario	Dependency Service	Dependency Policy	Supported Function
Real-time services	LTS	lts:logs:list (Obtaining the log list)	Show LTS logs.
	OBS	obs:bucket:GetBucketPolicy (Obtaining a bucket policy) obs:bucket:HeadBucket (Obtaining bucket metadata) obs:bucket:ListAllMyBuckets (Obtaining a bucket list) obs:bucket:PutBucketPolicy (Configuring a bucket policy) obs:bucket:DeleteBucketPolicy (Deleting a bucket policy)	Mount external volumes to a container when services are running.
Batch services	OBS	obs:object:GetObject (Obtaining object content and metadata) obs:object:PutObject (Uploading objects using PUT method, uploading objects using POST method, copying objects, appending an object, initializing a multipart task, uploading parts, and merging parts) obs:bucket:CreateBucket (Creating a bucket) obs:bucket:ListBucket (Listing objects in a bucket) obs:bucket:ListAllMyBuckets (Obtaining a bucket list)	Create batch services and perform batch inference.
Edge services	CES	ces:metricData:list: (Obtaining metric data)	View monitoring metrics.
	IEF	ief:deployment:delete (Deleting a deployment)	Manage edge services.
AOM metric alarm events	AOM	aom:alarm:list	View AOM monitoring information.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Table 10-13 Inference deployment (new version)

Scenario	Dependency Service	Dependent Policy	Supported Function
Real-time services	LTS	lts:groups:create lts:groups:get lts:groups:list lts:topics:create lts:topics:delete lts:topics:get lts:topics:list	Show LTS logs.
	OBS	obs:bucket:GetBucketLocation obs:bucket:HeadBucket obs:bucket:ListAllMybuckets obs:bucket:ListBucket obs:bucket:PutBucketAcl obs:object:AbortMultipartUpload obs:object:DeleteObject obs:object:DeleteObjectVersion obs:object:GetObject obs:object:GetObjectAcl obs:object:GetObjectVersion obs:object:GetObjectVersionAcl obs:object:ListMultipartUploadParts obs:object:PutObject obs:object:PutObjectAcl	Obtain bucket information.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Scenario	Dependency Service	Dependent Policy	Supported Function
	SWR	swr:instance:list swr:repo:listRepoTags swr:repository:getTag swr:repository:listArtifacts swr:repository:listRepositories swr:repo:download	Obtain enterprise images.
	CSMS	csms:secret:list csms:secretVersion:get	Obtain DEW secrets.
	SFS Turbo	sfsturbo:shares:getAllShares sfsturbo:shares:getShare sfsturbo:shares:showFsDir	Obtain SFS Turbo file systems.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Scenario	Dependency Service	Dependent Policy	Supported Function
	ModelArts	modelarts:authorization:list modelarts:clusterFlavor:list modelarts:image:get modelarts:image:list modelarts:image:listGroup modelarts:network:list modelarts:pool:get modelarts:pool:list modelarts:pool:update modelarts:service:create modelarts:service:get modelarts:service:list modelarts:tag:listTags modelarts:warmupTask:list modelarts:workspace:checkAuthorizations modelarts:workspace:get modelarts:workspace:getAuthMode modelarts:workspace:list	Obtains the service list. Create a real-time service. Obtains service tags. Authenticate workspaces. Obtain the public resource pool and dedicated resource pool information. Obtain image details. Obtain networks. Obtain the pre-warmed model list.
	VPC	vpc:vpcs:list vpc:subnets:get	Obtain network information.
	VPCEP	vpcep:epservices:create	Access the VPCEP information.
	KMS	kms:dek:decrypt	Decrypt a data key.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Table 10-14 Managing datasets

Application Scenario	Dependency Service	Dependency Policy	Supported Function
Managing datasets and labels	OBS	obs:bucket:GetBucketLocation obs:bucket:PutBucketAcl obs:object:PutObjectAcl obs:object:GetObjectVersion obs:object:GetObject obs:object:GetObjectVersionAcl obs:object:DeleteObject obs:object:ListMultipartUpload-Parts obs:bucket:HeadBucket obs:object:AbortMultipartUpload obs:object:DeleteObjectVersion obs:object:GetObjectAcl obs:bucket:ListAllMyBuckets obs:bucket:ListBucket obs:object:PutObject	Manage datasets in OBS. Label OBS data. Create a data management job.
Managing table datasets	DLI	dli:database:displayAllDatabases dli:database:displayAllTables dli:table:describeTable	Manage DLI data in a dataset.
Managing table datasets	DWS	dws:openAPICluster:list dws:openAPICluster:getDetail dws:cluster:list	Manage DWS data in a dataset.
Managing table datasets	MRS	mrs:job:submit mrs:job:list mrs:cluster:list mrs:cluster:get	Manage MRS data in a dataset.
Auto labeling	ModelArts	modelarts:service:list modelarts:model:list modelarts:model:get modelarts:model:create modelarts:trainJobInnerModel:list modelarts:workspace:get modelarts:workspace:list	Enable auto labeling.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Application Scenario	Dependency Service	Dependency Policy	Supported Function
Team labeling	IAM	iam:projects:listProjects (Obtaining tenant projects) iam:users:listUsers (Obtaining users) iam:agencies:createAgency (Creating an agency) iam:quotas:listQuotasForProject (Obtaining the quotas of a project)	Manage labeling teams.

Table 10-15 Managing resources

Application Scenario	Dependency Service	Dependency Policy	Supported Function
Managing resource pools	BSS	bss:coupon:view bss:order:view bss:balance:view bss:discount:view bss:renewal:view bss:bill:view bss:contract:update bss:order:pay bss:unsubscribe:update bss:renewal:update bss:order:update	Create, renew, and unsubscribe from a resource pool.
	CCE	cce:cluster:list cce:cluster:get	Obtain the CCE cluster list, cluster details, and cluster certificates.
	KMS	kms:cmk:list kms:cmk:getMaterial	Obtain the key pairs created by the user.
	AOM	aom:metric:get	Obtain the monitoring data of a resource pool.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Application Scenario	Dependency Service	Dependency Policy	Supported Function
	OBS	obs:bucket:ListAllMybuckets obs:bucket:HeadBucket obs:bucket:ListBucket obs:bucket:GetBucketLocation obs:object:GetObject obs:object:PutObject obs:object:DeleteObject obs:object:DeleteObjectVersion	Obtain AI diagnostic logs.
	ECS	ecs:availabilityZones:list ecs:cloudServerFlavors:get ecs:cloudServerQuotas:get ecs:quotas:get ecs:serverKeypairs:list	Obtain the AZs, specifications, and quotas, and configure keys.
	EVS	evs:types:get evs:quotas:get	Query EVS disk types and quotas.
	BMS	bms:serverFlavors:get	Query BMS specifications. Dependent permissions must be configured in the IAM project view.
	DEW	kps:domainKeypairs:list	Configure a key pair. Dependent permissions must be configured in the IAM project view.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Application Scenario	Dependency Service	Dependency Policy	Supported Function
Network management	VPC	vpc:routes:create vpc:routes:list vpc:routes:get vpc:routes:delete vpc:peerings:create vpc:peerings:accept vpc:peerings:get vpc:peerings:delete vpc:routeTables:update vpc:routeTables:get vpc:routeTables:list vpc:vpcs:create vpc:vpcs:list vpc:vpcs:get vpc:vpcs:delete vpc:subnets:create vpc:subnets:get vpc:subnets:delete vpcep:endpoints:list vpcep:endpoints:create vpcep:endpoints:delete vpcep:endpoints:get vpc:ports:create vpc:ports:get vpc:ports:update vpc:ports:delete vpc:networks:create vpc:networks:get vpc:networks:update vpc:networks:delete vpc:securityGroups:get	Create and delete ModelArts networks, and interconnect VPCs.
	SFS Turbo	sfsturbo:shares:addShareNic sfsturbo:shares:deleteShareNic sfsturbo:shares:showShareNic sfsturbo:shares:listShareNics	Interconnect your network with SFS Turbo.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Application Scenario	Dependency Service	Dependency Policy	Supported Function
	SWR	swr:instance:get swr:instance:createTempCredential swr:repository:getArtifact swr:repo:getRepoTag	Associate a network with an SWR enterprise repository.
Edge resource pool	IEF	ief:node:list ief:group:get ief:application:list ief:application:get ief:node:listNodeCert ief:node:get ief:IEFInstance:get ief:deployment:list ief:group:listGroupInstanceState ief:IEFInstance:list ief:deployment:get ief:group:list	Add, delete, modify, and search for edge pools.
Lite Cluster	ECS	ecs:cloudServers:get ecs:cloudServers:showServer ecs:cloudServers:changeVpc ecs:cloudServers:start ecs:cloudServers:listServerInterfaces ecs:cloudServers:delete ecs:cloudServers:redeploy ecs:cloudServers:batchSetServerTags ecs:cloudServers:reboot ecs:cloudServerFlavors:get ecs:cloudServers:list ecs:quotas:get ecs:cloudServers:create ecs:cloudServers:stop	Create, query, and delete cloud servers.
	APM	apm:icmgr:create	Install ICAgent.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Application Scenario	Dependency Service	Dependency Policy	Supported Function
	EVS	evs:volumes:list evs:types:get evs:volumes:get evs:quotas:get	Query EVS disk types, lists, quotas, and details.
	DEW	kps:domainKeypairs:get"	Configure a key pair. Dependency permissions must be configured in the IAM project view.
	BMS	bms:servers:create bms:serverFlavors:get bms:servers:redeploy	Query BMS specifications. Dependency permissions must be configured in the IAM project view.
	IMS	ims:images:get ims:images:share	Obtain details about an image and share the image.
	CCE	cce:node:delete cce:nodepool:delete cce:addonInstance:update cce:cluster:revokeClientCredential cce:node:get cce:addonInstance:list cce:node:create cce:node:list cce:addonInstance:get cce:accessPolicy:post cce:accessPolicy:list cce:accessPolicy:delete cce:node:remove cce:nodepool:create cce:addonInstance:delete cce:cluster:get cce:addonInstance:create cce:job:get	Obtain the CCE cluster list, cluster details, and cluster certificates.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Application Scenario	Dependency Service	Dependency Policy	Supported Function
	VPC	vpc:routeTables:get vpc:routeTables:list vpc:routes:create vpc:vpcs:get vpc:routes:get vpc:subnets:get vpc:routes:delete vpc:routes:list vpc:peerings:accept vpc:routeTables:update	Create and delete ModelArts networks, and interconnect VPCs.
	SFSTurbo	sfsturbo:shares:listShareNics sfsturbo:shares:deleteShareNic sfsturbo:shares:showShareNic sfsturbo:shares:addShareNic	Interconnect your network with SFS Turbo.
	SFSTurboReadOnly	SFS Turbo ReadOnlyAccess	Interconnect your network with SFS Turbo.

Table 10-16 Creating custom policies as an IAM user

Application Scenario	Dependency Service	Dependency Policy	Supported Function
Configuring fine-grained authorization	IAM	modelarts:authorization:list	Query fine-grained agency authorizations as an IAM user.
		modelarts:authorization:create	Create fine-grained agency authorizations as an IAM user.
		modelarts:authorization:delete	Delete fine-grained agency authorizations as an IAM user.

Agency authorization

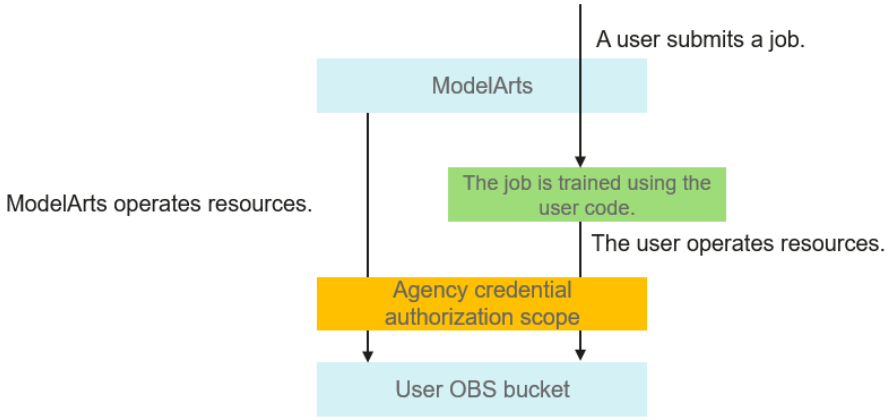
To simplify operations when you use ModelArts, certain operations are automatically performed on the ModelArts backend, for example, downloading the datasets in an OBS bucket to a workspace before a training job is started and dumping training job logs to the OBS bucket.

ModelArts does not save your token authentication credentials. Before performing operations on your resources (such as OBS buckets) in a backend asynchronous

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

job, you are required to explicitly authorize ModelArts through an IAM agency. ModelArts will use the agency to obtain a temporary authentication credential for performing operations on your resources. For details, see **Adding an Agency**.

Figure 10-4 Agency authorization



As shown in **Figure 10-4**, after authorization is configured on ModelArts, ModelArts uses the temporary credential to access and operate your resources, relieving you from some complex and time-consuming operations. The agency credential will also be synchronized to your jobs (including notebook instances and training jobs). You can use the agency credential to access your resources in the jobs.

You can use either of the following methods to authorize ModelArts using an agency:

1. One-click authorization

ModelArts provides one-click automatic authorization. You can quickly configure agency authorization on the **Permission Management** page of ModelArts. Then, ModelArts will automatically create an agency for you and configure it in ModelArts.

In this mode, the authorization scope is specified based on the preset system policies of dependency services to ensure sufficient permissions for using services. The created agency has almost all permissions of dependency services. If you want to precisely control the scope of permissions granted to an agency, use the second method.

2. Custom authorization

The administrator creates different agency authorization policies for different users in IAM, and configures the created agency for ModelArts users. When creating an agency for an IAM user, the administrator specifies the minimum permissions for the agency based on the user's permissions to control the resources that the user can access when they use ModelArts. For details, see **Assigning Basic Permissions for Using ModelArts**.

Risks in Unauthorized Operations

The agency authorization of a user is independent. Theoretically, the agency authorization scope of a user can be beyond the authorization scope of the

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

authorization policy configured for the user group. Any improper configuration will result in unauthorized operations.

To prevent unauthorized operations, only a tenant administrator is allowed to configure agencies for users to ensure the security of agency authorization.

Minimal Agency Authorization

When configuring agency authorization, an administrator must strictly control the authorization scope.

ModelArts asynchronously and automatically performs operations such as job preparation and clearing. The required agency authorization is within the basic authorization scope. If you use only some functions of ModelArts, the administrator can filter out the basic permissions that are not used according to the agency authorization configuration. Conversely, if you need to obtain resource permissions beyond the basic authorization scope in a job, the administrator can add new permissions to the agency authorization configuration. In a word, the agency authorization scope must be minimized and customized based on service requirements.

Basic Agency Authorization Scope

To customize the permissions for an agency, select permissions based on your service requirements.

Table 10-17 Basic agencies and authorizations in the development environment

Application Scenario	Dependency Service	Agency Authorization	Description
Performing operations on OBS data in a notebook instance	OBS	obs:object:DeleteObject obs:object:GetObject obs:object:GetObjectVersion obs:bucket:CreateBucket obs:bucket:ListBucket obs:bucket:ListAllMyBuckets obs:object:PutObject obs:bucket:GetBucketAcl obs:bucket:PutBucketAcl obs:bucket:PutBucketCORS	You can use either of the following methods to perform operations on OBS data in a notebook instance: • Use ModelArts SDK to perform operations on OBS data. • Use the notebook file upload function to perform operations on OBS data. • On the ModelArts console, add an OBS bucket to the **/data** directory of a notebook instance, and perform operations on OBS data in file mode.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Application Scenario	Dependency Service	Agency Authorization	Description
Reporting notebook instance events	AOM	aom:alarm:put	Some events during a notebook's lifecycle are reported to your AOM account. For details, see **Viewing Notebook Events**.
Interconnecting VPC with a notebook instance	VPC	vpc:ports:create vpc:ports:get vpc:ports:delete vpc:subnets:get	Add a NIC in the notebook instance for interconnecting with specified services in the VPC.
Connecting to a notebook instance through VS Code with one click	ModelArts	modelarts:notebook:get	Manage notebook instance details. Click **VS Code** to obtain the instance details and easily modify the instance information by writing the SSH configuration to the local VS Code.
Stopping a notebook instance	ModelArts	modelarts:notebook:stop	Stops a running notebook instance.
Updating the auto stop time of a notebook instance	ModelArts	modelarts:notebook:updateStopPolicy	Update the auto stop time of a notebook instance.
MindInsight/ TensorBoard used in OBS parallel file systems	ModelArts	modelarts:notebook:umountStorage modelarts:notebook:getMountedStorage modelarts:notebook:listMountedStorages modelarts:notebook:mountStorage	If MindInsight or TensorBoard is enabled in a notebook instance, and you need to access the OBS parallel file system, configure the permissions on the left.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Application Scenario	Dependency Service	Agency Authorization	Description
Mounting SFS Turbo when a notebook instance uses a dedicated resource pool	SFS	sfsturbo:shares:getShare sfsturbo:shares:showFsDir	When creating a notebook instance that uses a dedicated resource pool, select SFS Turbo as the storage and verify the storage.

Table 10-18 Basic agency authorization for training jobs

Application Scenario	Dependency Service	Agency Authorization	Description
Accessing OBS files for training jobs	OBS	obs:bucket:HeadBucket obs:bucket:GetBucketLocation obs:bucket:ListBucket obs:bucket:ListAllMyBuckets obs:bucket:ListBucketVersions obs:object:GetObject obs:object:GetObjectVersion obs:object:GetObjectAcl obs:object:GetObjectVersionAcl	You need to obtain OBS operation permissions when configuring a training job, including the code directory, input, output, and the OBS bucket path for storing logs.
Starting a training job using a custom container image.	SWR	swr:repo:getRepo swr:system:createLoginSecret swr:instance:createTempCredential	When a training job is started using a custom container image, you need to obtain a temporary login command of the SWR container image to download the container image.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Application Scenario	Dependency Service	Agency Authorization	Description
Notification of training job status changes	SMN	smn:template:list smn:template:create smn:topic:list smn:topic:publish	To configure training job status change notifications, you must have the SMN operation permissions to send template-based notifications.
Mounting SFS Turbo to a training job	SFS Turbo	SFS Turbo ReadOnlyAccess	To mount SFS Turbo to a training job, you must have the SFS Turbo read permission to obtain its details by ID.

Table 10-19 Basic agency authorization for inference deployment (old version)

Application Scenario	Dependency Service	Agency Authorization	Description
Real-time services	LTS	lts:groups:create lts:groups:list lts:topics:create lts:topics:delete lts:topics:list	Configure LTS for reporting logs of real-time services.
Batch services	OBS	obs:bucket:ListBucket obs:object:GetObject obs:object:PutObject	This parameter is mandatory when a batch service is used.
Edge services	IEF	ief:deployment:list ief:deployment:create ief:deployment:update ief:deployment:delete ief:node:createNodeCert ief:iefInstance:list ief:node:list	This parameter is mandatory when an edge service is used. The edge service is deployed through IEF.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Application Scenario	Dependency Service	Agency Authorization	Description
Importing a model from OBS	OBS	obs:object:DeleteObject obs:object:GetObject obs:bucket:CreateBucket obs:bucket:ListBucket obs:object:PutObject obs:bucket:GetBucketAcl obs:bucket:PutBucketAcl obs:bucket:PutBucketCORS	(Mandatory) If a parallel file system is used, you need to configure obs:bucket:HeadBucket.
Importing a model from the container image	SWR	swr:instance:get swr:repository:getArtifact swr:repo:getRepoTag swr:repo:getRepo swr:repo:getRepoDomain swr:instance:createTempCredential swr:system:createLoginSecret swr:repo:deleteRepoTag swr:repo:deleteRepo swr:repo:createRepoDomain swr:repo:deleteRepoDomain	(Mandatory)
Using ModelArts Edge	IEF	ief:deployment:list ief:deployment:create ief:deployment:update ief:deployment:delete ief:node:createNodeCert ief:iefInstance:list ief:node:list	(Optional) This function must be enabled if ModelArts Edge is used.
AOM metric alarm events	AOM	aom:log:get aom:alarm:get aom:metric:put aom:alarm:put aom:event:put aom:event:list aom:event:get	Enable this function to view alarms and events on AOM.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Application Scenario	Dependency Service	Agency Authorization	Description
Reporting monitoring metrics to Cloud Eye	Cloud Eye	ces:metricMeta:create	Enable this function to report monitoring metrics to Cloud Eye.
Message subscription and push	SMN	smn:topic:list smn:topic:publish smn:application:publish	(Optional) Enable this function for message subscription and push.

Table 10-20 Basic agency authorization for inference deployment (new version)

Scenario	Dependent Service	Agency Authorization	Description
Real-time services	LTS	lts:groups:create lts:groups:get lts:groups:list lts:topics:create lts:topics:delete lts:topics:get lts:topics:list	Configure LTS for reporting logs of real-time services.
	SWR	swr:repo:download	Download SWR images.
	SFS Turbo	sfsturbo:shares:getShare sfsturbo:shares:showFsDir	Access SFS Turbo file systems.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Scenario	Dependent Service	Agency Authorization	Description
	OBS	obs:object:DeleteObject obs:object:GetObject obs:bucket:CreateBucket obs:bucket:ListBucket obs:object:PutObject obs:bucket:GetBucketAcl obs:bucket:PutBucketAcl obs:bucket:PutBucketCORS	Access OBS model data.
	CSMS	csms:secretVersion:get	Access the latest secret version.
	ModelArts	modelarts:authorization:list modelarts:image:get modelarts:pool:get modelarts:pool:list modelarts:pool:update modelarts:warmupTask:list	Obtain image details. Obtain details about a resource pool. Obtain details about a workspace.
	VPC	vpc:vpcs:list vpc:subnets:get	Access network information.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Scenario	Dependent Service	Agency Authorization	Description
	VPCEP	vpcep:epservices:delete vpcep:epservices:get vpcep:connections:update vpcep:permissions:update vpcep:endpoints:create vpcep:endpoints:delete vpcep:endpoints:get vpcep:endpoints:list	Access the VPCEP information.
	KMS	kms:dek:decrypt	Decrypt a data key.

Table 10-21 Basic agency authorization for managing data

Application Scenario	Dependency Service	Agency Authorization	Description
Data labeling and processing	ModelArts	modelarts:trainJob:create modelarts:trainJob:update modelarts:trainJob:delete modelarts:trainJob:get modelarts:trainJob:list modelarts:trainJob:logExport modelarts:aiAlgorithm:get modelarts:model:get modelarts:service:list modelarts:model:create modelarts:workspace:list modelarts:workspace:get modelarts:trainJobInnerModel:list	(Mandatory) Create and query training jobs, as well as querying algorithms.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Application Scenario	Dependency Service	Agency Authorization	Description
Accessing OBS data	OBS	obs:bucket:GetBucketLocation obs:bucket:PutBucketAcl obs:object:PutObjectAcl obs:object:GetObjectVersion obs:object:GetObject obs:object:GetObjectVersionAcl obs:object:DeleteObject obs:object:ListMultipartUpload-Parts obs:bucket:HeadBucket obs:object:AbortMultipartUpload obs:object:DeleteObjectVersion obs:object:GetObjectAcl obs:bucket:ListAllMyBuckets obs:bucket:ListBucket obs:object:PutObject	(Mandatory) Store, query, and delete data in OBS.
Accessing DLI data	DLI	dli:queue:createQueue dli:queue:dropQueue dli:queue:scaleQueue dli:queue:submitJob dli:database:displayDatabase dli:database:displayAllTables dli:table:describeTable dli:table:showPrivileges dli:table:dropTable	(Optional) Enable this function if you need to view the DLI data.
Accessing MRS data	MRS	mrs:job:submit mrs:job:list mrs:cluster:list mrs:file:list	(Optional) Enable this function if you need to view the MRS data.
Accessing DWS data	DWS	dws:openAPICluster:list dws:openAPICluster:getDetail dws:cluster:list	(Optional) Enable this function if you need to view the DWS data.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Table 10-22 Basic agency authorization for managing dedicated resource pools

Application Scenario	Dependency Service	Agency Authorization	Description
Interconnecting a dedicated resource pool with SFS Turbo resources	SFS Turbo	sfsturbo:shares:showShareNic sfsturbo:shares:listShareNics sfsturbo:shares:addShareNic sfsturbo:shares:deleteShareNic	Enable this function as needed.
Interconnecting ModelArts network with VPC and adding related routes	VPC	vpc:vpcs:get vpc:subnets:get vpc:peerings:accept vpc:routes:create vpc:routes:delete vpc:routes:get vpc:routeTables:update vpc:routeTables:get vpc:routeTables:list vpc:routes:list	Enable this function as needed.
Using ModelArts Lite Cluster resource pools	CCE APM	cce:cluster:get cce:node:get cce:node:list cce:job:get cce:node:create cce:node:delete cce:node:remove cce:addonInstance:get cce:addonInstance:list cce:addonInstance:create cce:addonInstance:update cce:addonInstance:delete cce:accessPolicy:post cce:accessPolicy:list cce:accessPolicy:delete cce:cluster:revokeClientCredential cce:nodepool:create cce:nodepool:delete apm:icmgr:create	Must be configured when a ModelArts Lite Cluster resource pool is used. ModelArts uses an agency to manage CCE clusters, synchronize cluster information, and manage nodes.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Application Scenario	Dependency Service	Agency Authorization	Description
	ECS BMS EVS DEW	ecs:cloudServers:create ecs:cloudServers:delete ecs:cloudServers:get ecs:cloudServers:start ecs:cloudServers:stop ecs:cloudServers:reboot ecs:cloudServers:redeploy ecs:cloudServers:listServerInterfaces ecs:cloudServers:changeVpc ecs:cloudServerFlavors:get ecs:quotas:get ecs:cloudServers:batchSetServerTags ecs:cloudServers:list bms:servers:create bms:serverFlavors:get evs:types:get evs:volumes:list evs:quotas:get evs:volumes:get kps:domainKeypairs:get	Must be configured when a ModelArts Lite Cluster resource pool is used. ModelArts uses an agency to manage the lifecycle of BMSs and ECSs.
	IMS	ims:images:get ims:images:share	Must be configured when a ModelArts Lite Cluster resource pool is used. Share the node system image with your account before creating a ModelArts Lite Cluster resource pool node.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Application Scenario	Dependency Service	Agency Authorization	Description
	CloudMatrix	cloudmatrix:hyperinstanceCluster:create cloudmatrix:hyperinstanceCluster:list cloudmatrix:hyperinstanceCluster:get cloudmatrix:hyperinstanceCluster:delete cloudmatrix:hyperinstanceCluster:getClusterSpec cloudmatrix:hyperinstance:create cloudmatrix:hyperinstance:delete cloudmatrix:hyperinstance:get cloudmatrix:hyperinstance:list cloudmatrix:hyperinstance:listNodes cloudmatrix:hyperinstance:getNode cloudmatrix:hyperinstance:stopNode cloudmatrix:hyperinstance:rebootNode cloudmatrix:hyperinstance:startNode cloudmatrix:hyperinstance:changeOs cloudmatrix:hyperinstance:reinstallNodeOs cloudmatrix:hyperinstance:liveScaleUp cloudmatrix:hyperinstance:liveScaleDown cloudmatrix:operation:get cloudmatrix:operation:list cloudmatrix:scheduledEvent:create cloudmatrix:scheduledEvent:list cloudmatrix:scheduledEvent:accept cloudmatrix:scheduledEvent:cancel cloudmatrix:hyperinstance:detachNodeInterface	Must be configured when a ModelArts Lite Cluster resource pool uses CloudMatrix supernodes. ModelArts uses agencies to manage the lifecycle of CloudMatrix supernodes.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

10.2.3 Workspace

ModelArts allows you to create multiple workspaces to develop algorithms and manage and deploy models for different service objectives. In this way, the development outputs of different applications are allocated to different workspaces for simplified management.

Workspace supports the following types of access control:

- **PUBLIC**: publicly accessible to tenants (including both the tenant account and all its IAM accounts)
- **PRIVATE**: accessible only to the creator and the tenant account
- **INTERNAL**: accessible to the creator, the tenant account, and specified IAM users. When **Authorization Type** is set to **INTERNAL**, you need to specify one or more accessible IAM users.

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

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

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

NOTE

The permission **modelarts:workspace:list** only works if a resource space is set up. It cannot be used with enterprise projects.

10.3 Configuring Agency Authorization for ModelArts with One Click

Scenario

ModelArts must access other services for AI computing. For example, ModelArts must access OBS to read your data for training. To ensure security, ModelArts requires you to configure an agency before accessing any cloud services. This process involves delegating permissions to ModelArts so it can access those services on your behalf. You must first have the required permissions (granted by IAM), and then delegate ModelArts to access specific cloud services on your behalf to complete AI computing tasks on the ModelArts platform.

ModelArts allows you to quickly add an agency on the **Permission Management** page. This enables ModelArts to manage service-to-service permissions on your behalf.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Constraints

- The user group to which the user belongs has been authorized using IAM.
- Huawei Cloud account
  - Only a Huawei Cloud account can use an agency to authorize the current account or all IAM users under the current account.
  - Multiple IAM users or accounts can use the same agency.
  - A maximum of 100 agencies can be created under an account.
  - If you use ModelArts for the first time, add an agency. Generally, common user permissions are sufficient for your requirements. You can configure permissions for refined permissions management. On the new ModelArts console, new users can configure ModelArts agency authorization for their accounts in one-click mode.
- IAM user
  - If you have obtained the authorization, you can view the authorization information on the **Permission Management** page.
  - If you have not been authorized, ModelArts will display a message indicating that you have not been authorized when you access the **Add Agency** page. In this case, contact your administrator to add authorization.
  - An IAM user with the admin permission can add agencies for itself or other IAM users.

One-Click Authorization

When you use ModelArts for the first time with a Huawei Cloud account or an IAM admin user, a cross-service authorization page appears on the new ModelArts console. Click the authorize button to set up ModelArts authorization quickly.

NOTE

One-click authorization is only supported on the new ModelArts console in the following regions:

ME-Riyadh

Adding an Agency

If you are not using the ModelArts new console for the first time or you are using the ModelArts old console, you can add an agency on the **Permission Management** page.

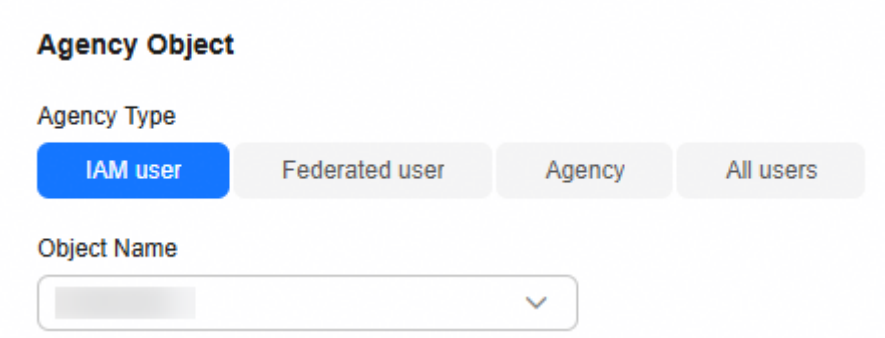
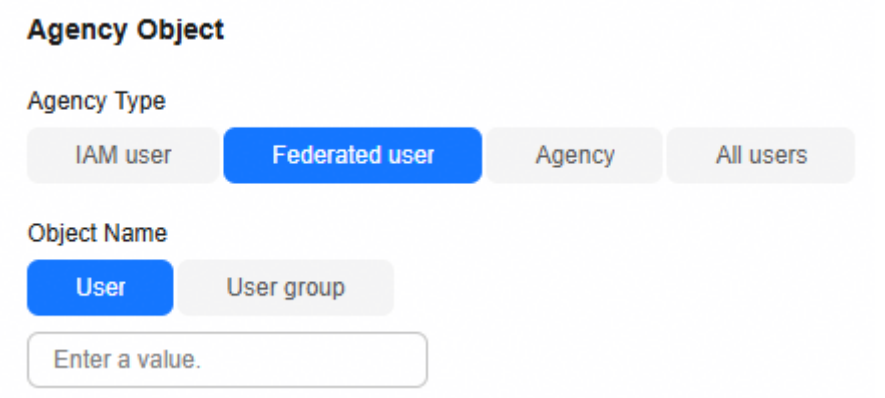
1. Log in to the **ModelArts console**. In the navigation pane on the left, choose **Permission Management**.
2. Click **Add Agency**. On the **Add Agency** page, configure the parameters.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

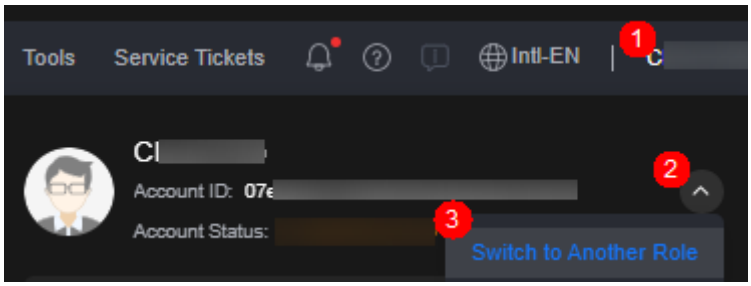
Table 10-23 Parameters

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

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Parameter	Description
Object Name	This parameter is not displayed when **Agency Type** is set to **All users**. It is displayed when **Agency Type** is set to **IAM users**: • **IAM user**: Select an IAM user and configure an agency for the IAM user. **Figure 10-5** Selecting an IAM user  • When **Agency Type** is set **Federated users**: The object name can be a federated user or user group. Enter the username or user ID of the federated user. **Figure 10-6** Selecting a federated user or user group  ModelArts only supports federated users of the virtual user SSO type. This restriction also applies to federated user groups. Permissions granted to a federated user group take effect only for AI jobs. To manage resource pools, the resource administrator must log in to the console using IAM. Add a federated user to only one user group to use the authorization feature. Adding a federated user to multiple groups can cause random authorization effects. When a federated user switches to a new user group, their permissions on ModelArts match those of the new group. After changing user groups for a notebook instance, restart the instance to access or edit notebook instances from the old group.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Parameter	Description
	• **Agency**: Select an agency name. You can create an agency under account A and grant the agency permissions to account B. When using account B, you can switch the role in the upper right corner of the console to account A and use the agency permissions of account A. **Figure 10-7** Switch Role 
Agency Type	• **Add agency**: If there is no available agency, create one. If you use ModelArts for the first time, select **Add agency**. • **Use existing**: If there are agencies in the list, select an available one to authorize the selected user. Click the drop-down arrow next to the agency name to view its permission details.
Add agency > Agency Name	The system automatically creates an editable agency name. ModelArts creates agency names using these rules: 1. When the agency object type is IAM user, the default agency name is ma_agency_[object_name]. 2. For other agency object types, the default agency name is modelarts_agency. 3. If an agency name generated based on preceding rules already exists, a four-digit random code is automatically suffixed to the name.
Add agency > Agency Configuration	Users can customize the authorization scope based on service scenarios and grant only the minimum required access. This mode is flexible and secure. This mode applies to scenarios with refined management requirements on the authorization scope. You can select the required permission from the list. • Agency template: ModelArts provides preset agency templates for quick and easy configuration. For example, when selecting the agency template for the training module, the system automatically chooses all necessary service and configuration permissions. • Service list: You can select services from the list and view and configure the required permissions in the configuration details. • Configuration details: View and configure the required permissions.

3. Click **Create**, enter **YES** as prompted, and click **OK**.

   A message will then appear confirming that the agency has been created.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

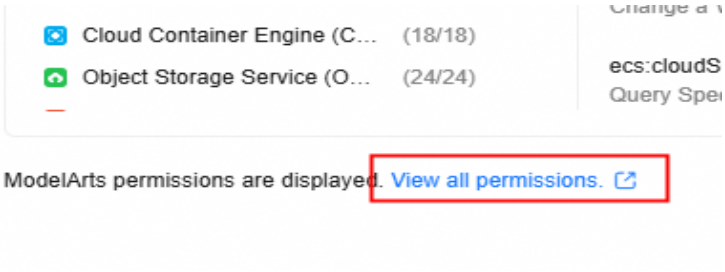
Viewing the Agency List

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

Modifying the Authorization Scope

1. On the agency list page, click **View Permissions** to view the permissions of the agency.
2. To modify the authorization scope, click **View all permissions** at the bottom of the **View Permissions** dialog box.

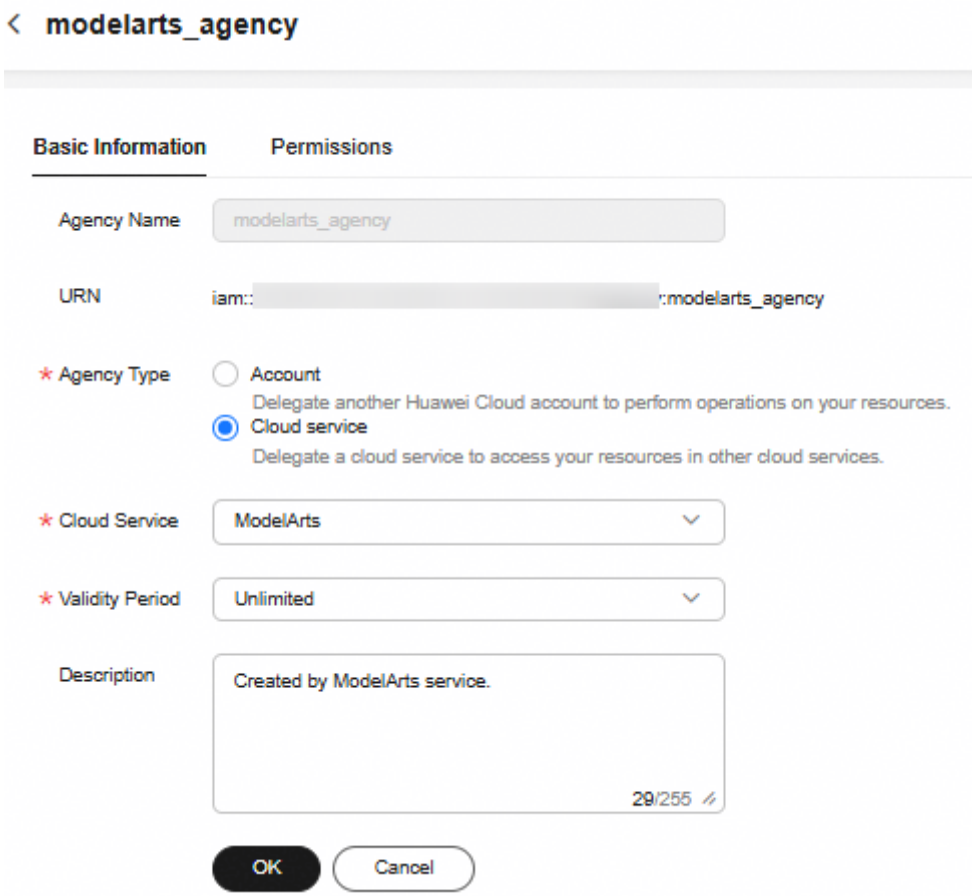
   Figure 10-8 Modifying permissions in IAM

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

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Figure 10-9 Agency information



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

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

Deleting an Authorization

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

- **Deleting an agency of a user**

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

- **Batch deleting all agencies**

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

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

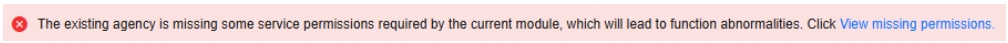
FAQs

1. How do I configure authorization when I use ModelArts for the first time?

   When you use ModelArts for the first time with a tenant account (administrator) or an IAM admin user, a cross-service authorization page appears on the new ModelArts console. Click the authorize button to set up ModelArts authorization quickly. Other IAM users need to be granted permissions by the administrator. For details, see **Creating an IAM User and Granting ModelArts Permissions**. When the tenant account (administrator) or an IAM user with the admin permission accesses the ModelArts console of the old version, follow the instructions in **Adding an Agency**.

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

   Figure 10-10 Insufficient permissions

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

10.4 Creating an IAM User and Granting ModelArts Permissions

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

ModelArts allows you to configure fine-grained permissions for refined management of resources and permissions. This type of feature is commonly used in scenarios with large-scale enterprise users. If you want to precisely control the scope of permissions granted to an agency, use custom authorization.

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

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

- For details about how to grant operation permissions of ModelArts and dependent services to a user, see **Configuring Common Operations Permissions for ModelArts**.
- For details about how to perform refined management of user permissions on ModelArts and dependent services and how to configure custom policies, see **Creating a Custom Policy for ModelArts**.

Prerequisites

You have learned about the permissions that can be added to user groups for using ModelArts and dependent services so that you can select them based on your requirements. For details about the permissions supported by ModelArts, see **Table 10-24**.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Table 10-24 Service authorization

Target Service	Authorization Description	IAM Permission	Mandatory
ModelArts	This permission allows IAM users to use ModelArts. The users with the ModelArts CommonOperations permission can only use resources, but cannot create, update, or delete any dedicated resource pool. You are advised to assign this permission to IAM users.	ModelArts CommonOperations	Yes
	The IAM users with the **ModelArts FullAccess** permission have all access permissions, including creating, updating, and deleting dedicated resource pools. Exercise caution when assigning this permission.	ModelArts FullAccess	No Select either **ModelArts FullAccess** or **ModelArts CommonOperations**.
Object Storage Service (OBS)	This permission allows IAM users to use OBS. ModelArts datasets, development environments, training jobs, and model inference and deployment require **OBS for forwarding data**.	OBS OperateAccess	Yes
Software Repository for Container (SWR)	This permission allows IAM users to use SWR. ModelArts custom images require the SWR FullAccess permission.	SWR OperateAccess	Yes

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Target Service	Authorization Description	IAM Permission	Mandatory
Key Management Service (KMS)	This permission allows IAM users to use remote SSH of ModelArts notebook. KMS usage permission needs to be configured.	DEW KeypairFullAccess KMS CMKFullAccess	No
Intelligent EdgeFabric (IEF)	This permission allows IAM users to use IEF. **Tenant Administrator** permission is required if you use ModelArts edge services that depend on IEF.	Tenant Administrator	No
Cloud Eye	This permission allows IAM users to use Cloud Eye. Using Cloud Eye, you can view the running statuses of ModelArts real-time services and AI application loads, and set monitoring alarms.	CES FullAccess	No
Simple Message Notification (SMN)	This permission allows IAM users to use SMN. SMN is used with Cloud Eye.	SMN FullAccess	No
Virtual Private Cloud (VPC)	When creating a ModelArts dedicated resource pool, IAM users require VPC permissions so that they can customize networks.	VPC FullAccess	No
Scalable File Service (SFS)	This permission allows IAM users to use SFS. SFS file systems can be mounted to ModelArts dedicated resource pools to serve as storage for development environments or training jobs.	SFS Turbo FullAccess SFS FullAccess	No

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Configuring Common Operations Permissions for ModelArts

Step 1 Create a user group.

Log in to the IAM console. Choose **User Groups** and click **Create User Group**. Enter a user group name, and click **OK**.

Step 2 Configure permissions for the user group.

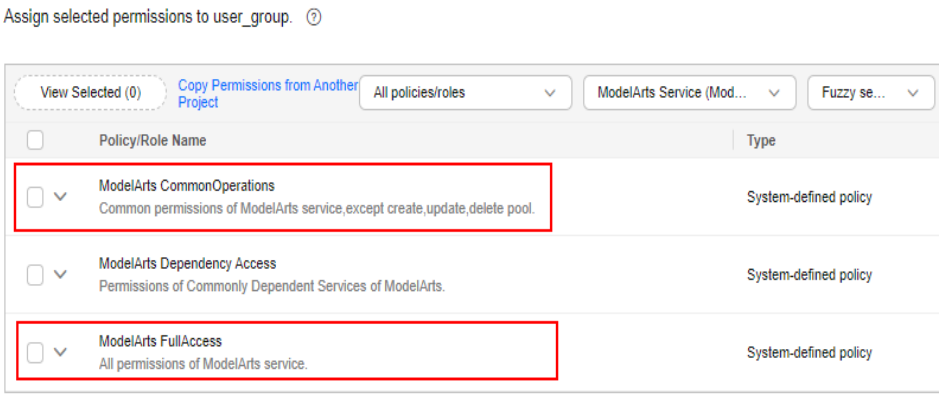
In the user group list, locate the created user group, click **Authorize** in the **Operation** column, and perform the following operations.

1. Assign permissions for using ModelArts. Search for **ModelArts** in the search box. Select either **ModelArts FullAccess** or **ModelArts CommonOperations**.

   The differences between the options are as follows:

   - The users with the ModelArts CommonOperations permission can only use resources, but cannot create, update, or delete any dedicated resource pool. You are advised to assign this permission to IAM users.
   - The IAM users with the **ModelArts FullAccess** permission have all access permissions, including creating, updating, and deleting dedicated resource pools. Exercise caution when assigning this permission.

   Figure 10-11 Assigning permissions for using ModelArts

2. Configure permissions for using other dependency services. For example, to use OBS, search for **OBS** and select **OBS OperateAccess**. ModelArts training jobs use OBS to forward data. Therefore, the permissions for using OBS are necessary.

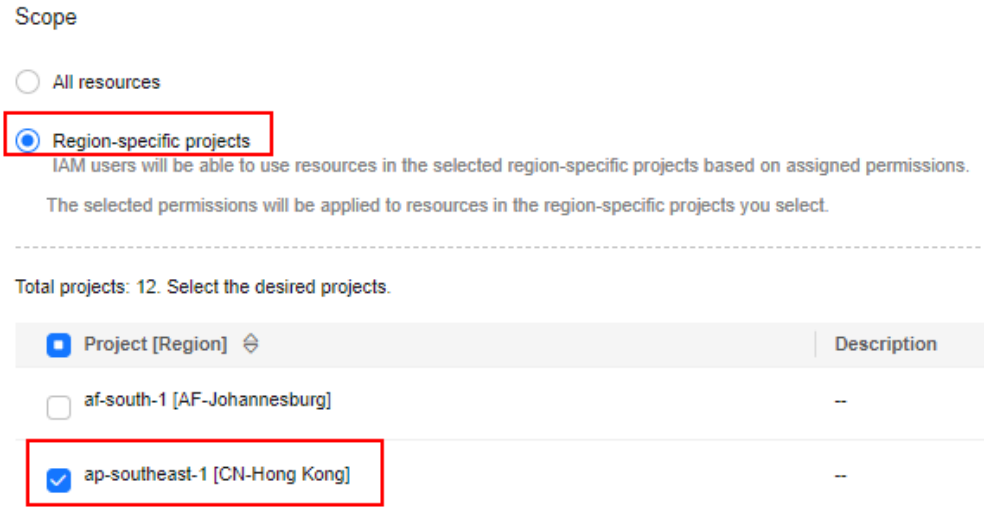
   For permissions of more cloud services, such as SWR, see **Table 10-24**.

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

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

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Figure 10-12 Selecting an authorization scope



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

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

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

Step 5 **Log in as the IAM user** and verify permissions.

In the authorized region, perform the following operations:

- Choose **Service List** > **ModelArts**. In the navigation pane of the ModelArts console, choose your desired type of AI dedicated resource pools and create one. You should not be able to create a new resource pool if the **ModelArts CommonOperations** permission has taken effect.
- Choose any other service in **Service List**. (Assume that the current policy contains only **ModelArts CommonOperations**.) If a message appears indicating that you have insufficient permissions to access the service, the **ModelArts CommonOperations** permission has taken effect.
- Choose **Service List** > **ModelArts**. In the navigation pane of the ModelArts console, choose **Asset Management** > **Datasets** and click **Create**. You should be able to access the corresponding OBS path if the OBS permission has taken effect.
- Verify other optional permissions according to **Table 10-24**.

----End

Creating a Custom Policy for ModelArts

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

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

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

For more about other functions and related services, see **Role/Policy-based Authorization**.

For details about how to create custom policies and related parameters, see **Creating a Custom Policy**.

NOTE

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

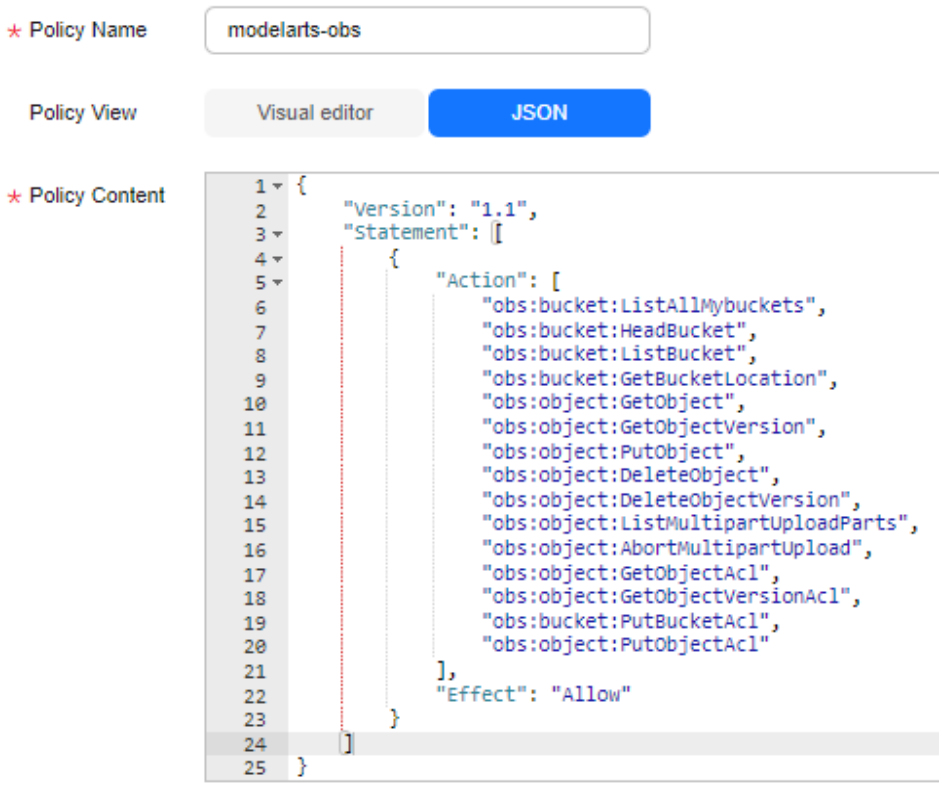
ModelArts is accessible through region-specific projects, but OBS is globally accessible, so you need to create separate policies for the two services and then apply these policies to the users.

1. Create a custom policy with minimum OBS permissions for ModelArts.

   Log in to the IAM console, choose **Permissions** > **Policies/Roles**, and click **Create Custom Policy**. Set the following parameters:

   - **Policy Name**: Enter a policy name.
   - **Policy View**: Select **JSON**.
   - **Policy Content**: Configure the policy by referring to **Example Custom Policy with OBS Permissions for ModelArts**.

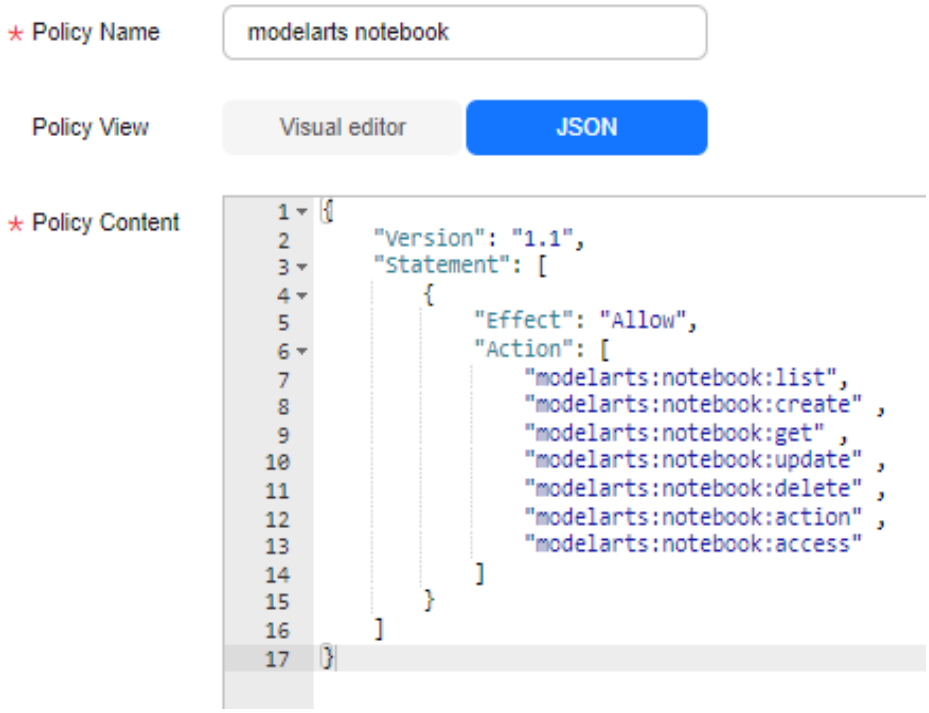
   Figure 10-13 Minimum OBS permissions



 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

2. Create a custom policy for the permission to use the ModelArts development environment, as shown in **Figure 10-14**. Set the following parameters:
   - **Policy Name**: Enter a policy name.
   - **Policy View**: Select **JSON**.
   - **Policy Content**: Follow the instructions in **Example Custom Policy with Permissions for Using the ModelArts Development Environment**. For the actions that can be added for custom policies, see **ModelArts API Reference > Permissions Policies and Supported Actions**.

   Figure 10-14 Permission to use the development environment



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

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Example Custom Policy with OBS Permissions for ModelArts

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

```
{
   "Version": "1.1",
   "Statement": [
      {
         "Action": [
            "obs:bucket:ListAllMybuckets",
            "obs:bucket:HeadBucket",
            "obs:bucket:ListBucket",
            "obs:bucket:GetBucketLocation",
            "obs:object:GetObject",
            "obs:object:GetObjectVersion",
            "obs:object:PutObject",
            "obs:object:DeleteObject",
            "obs:object:DeleteObjectVersion",
            "obs:object:ListMultipartUploadParts",
            "obs:object:AbortMultipartUpload",
            "obs:object:GetObjectAcl",
            "obs:object:GetObjectVersionAcl",
            "obs:bucket:PutBucketAcl",
            "obs:object:PutObjectAcl"
         ],
         "Effect": "Allow"
      }
   ]
}
```

Example Custom Policy with Permissions for Using the ModelArts Development Environment

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

Related References

Here are some other permission configuration examples for your reference.

-
- **Assigning the Permissions to Manage All Notebook Instances to an IAM User**
- **Separately Assigning Permissions to Administrators and Developers**
- **Prohibiting a User from Using a Public Resource Pool**

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

- **Assigning SFS Turbo Folder-Level Access Permissions to an IAM User**

10.5 Creating and Managing Workspaces

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

Workspace supports the following types of access control:

- **PUBLIC**: publicly accessible to tenants (including both the tenant account and all its IAM users)
- **PRIVATE**: accessible only to the creator and the tenant account
- **INTERNAL**: accessible to the creator, the tenant account, and specified IAM users. When the authorization type is set to **INTERNAL**, specify one or more accessible IAM users.

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

Application Scenarios

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

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

Prerequisites

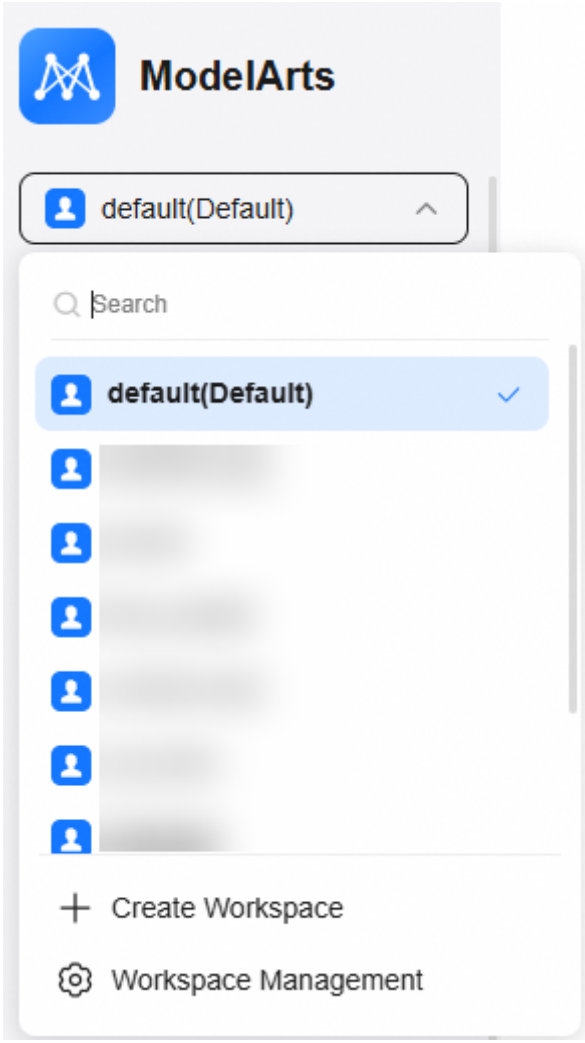
You have configured the basic permissions for using ModelArts and IAM. For details about ModelArts permission configurations, see **Configuring Agency Authorization for ModelArts with One Click**. For details about IAM permission configurations, see **Workspace Management**.

Choosing a Workspace

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

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Figure 10-15 Switching the Workspace



Creating a Workspace

1. Log in to the **ModelArts console**.
2. In the workspace drop-down list in the navigation pane on the left, click **Create Workspace**. The page for creating a workspace is displayed.

Table 10-25 Creating a workspace

Parameter	Description
Workspace Name	Enter the name of the workspace. This parameter is mandatory. The value can contain 4 to 64 visible characters, including letters, digits, hyphens (-), and underscores (_).

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Parameter	Description
Description (Optional)	Enter the description of the workspace. The description can contain 0 to 256 characters. Only letters, digits, hyphens (-), underscores (_), commas (,), periods (.), and spaces are allowed.
Enterprise Project	Select an enterprise project. This parameter is mandatory. If no enterprise project is available, click **Create Enterprise Project** to go to the **Enterprise Project Management** page, create an enterprise project, and bind it to the workspace. Enterprise projects provide a cloud resource management mode, in which cloud resources and members are centrally managed by project.
Authorization Type	Select the access control mode for the workspace. • **PUBLIC**: publicly accessible to tenants (including both the tenant account and all its IAM users) • **PRIVATE**: accessible only to the creator and the tenant account • **INTERNAL**: accessible to the creator, the tenant account, and specified IAM users. When **Authorization Type** is set to **INTERNAL**, you need to set **Authorized User** and **Authorized To** to specify the IAM users who can access the workspace. When **Authorized User** is set to **IAM user**, select one or more IAM usernames in **Authorized To**. When **Authorized User** is set to **Agency**, select one or more agency names in **Authorized To**. When **Authorized User** is set to **Federated user**, enter one or more user IDs of federated users in **Authorized To**. When **Authorized User** is set to **Federated user group**, select one or more user group names in **Authorized To**.

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

Managing Workspace Quotas

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

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

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

4. Click **Modify Quota** on the right of **Quota Info** to change the quota value. For details about the configurations, see **Table 10-26**.

Table 10-26 Quota information

Quota	Value Description	Unit
GPU-based training job duration (calculated based on a single Pnt1 per node)	The value ranges from **1** to **60000** and is unlimited by default.	Minute
CPU-based training job duration (calculated based on a single vCPU per node)	The value ranges from **1** to **60000** and is unlimited by default.	Minute
Visualization job duration	The value ranges from **1** to **60000** and is unlimited by default.	Minute
CPU-based development environment duration (calculated based on the number of vCPUs)	The value ranges from **1** to **60000** and is unlimited by default.	Minute
GPU-based development environment duration (calculated based on the number of Pnt1 GPUs)	The value ranges from **1** to **60000** and is unlimited by default.	Minute
CPU-based inference service duration (calculated based on the number of nodes)	The value ranges from **1** to **60000** and is unlimited by default.	Minute
GPU-based inference service duration (calculated based on the number of nodes)	The value ranges from **1** to **60000** and is unlimited by default.	Minute
vCPUs for CPU-based training jobs	The value ranges from **1** to **10000** and is unlimited by default.	vCPU
Processing units (PUs) for GPU-based training jobs	The value ranges from **1** to **1000** and is unlimited by default.	PU
RAM size for training jobs	The value ranges from **1** to **100000** and is unlimited by default.	GB

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Quota	Value Description	Unit
GPU-based auto labeling duration	The value ranges from **1** to **60000** and is unlimited by default.	Minute

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

Modifying workspaces

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

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

   You can modify the name, description, and authorization type of the workspace. For details, see **Table 10-25**.
3. After modifying the parameters, click **Modify Now**.

Deleting workspaces

You can delete a workspace that is no longer needed. However, you need to delete all resources in the workspace first, including created notebooks, training jobs, deployment services, and associated resource pools and networks.

NOTE

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

1. In the navigation pane of the **ModelArts console**, choose **Workspace Management** from the workspace down-down list.
2. In the workspace list, click **Delete** in the **Operation** column. In the displayed dialog box, confirm the information about the workspace you want to delete.

   If there are resource pools and networks that have not been deleted in the workspace, delete them first.

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

Using CTS to Audit ModelArts Workspaces

ModelArts workspaces can connect to CTS. With CTS, you can obtain operations associated with ModelArts for later query, audit, and backtrack operations.

After you enable CTS, it automatically records operations and changes on all cloud resources under your account and saves the information as traces for seven days. This section describes how to view operation records of the last seven days on the CTS console.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Prerequisites

CTS has been enabled. For details, see **Cloud Trace Service User Guide**.

Key Workspace Operations That Can Be Audited

For details about the key workspace operations that can be recorded on CTS, see **Table 10-27**.

The **enterprise_project_id** field is added to the **CTS trace structure**. For details, see the ID of the enterprise project to which the resource belongs. In the following table, **EPS Support** indicates whether the operation supports the reporting of enterprise project IDs.

Table 10-27 Key workspace operations that can be audited by CTS

Operation	Resource Type	Trace	EPS Support
Creating a Workspace	workspace	createWorkspace	Yes
Modifying workspaces	workspace	updateWorkspace	Yes
Deleting workspaces	workspace	deleteWorkspace	Yes
Modifying workspace quotas	workspaceQuota	updateWorkspaceQuota	Yes

Procedure

1. Log in to the **CTS console**.
2. Click  in the upper left corner and select the target region.
3. In the navigation pane on the left, choose **Trace List**.
4. Filter traces to view information about the target traces.
   For details, see **Querying Traces in CTS**.

10.6 Configuration Practices in Typical Scenarios

10.6.1 Assigning Basic Permissions for Using ModelArts

10.6.1.1 Scenario

Certain ModelArts functions require the permission to access other services. This section describes how to assign specific permissions to IAM users when they use ModelArts.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Permissions

The permissions of IAM users are controlled by their tenant user. Logging in as a tenant user, you can assign permissions to the target user group through IAM. Then, the permissions are assigned to all members in the user group. The following authorization list uses the system-defined policies of ModelArts and other services as an example.

Table 10-28 Service authorization

Target Service	Description	IAM Permission	Mandatory
ModelArts	Assign permissions to IAM users for using ModelArts. The users with the ModelArts CommonOperations permission can only use resources, but cannot create, update, or delete any dedicated resource pool. You are advised to assign this permission to IAM users.	ModelArts CommonOperations	Yes
	The users with the ModelArts FullAccess permission have all access permissions, including creating, updating, and deleting dedicated resource pools. Exercise caution when selecting this option.	ModelArts FullAccess	No Select either **ModelArts FullAccess** or **ModelArts CommonOperations**.
Object Storage Service (OBS)	Assign permissions to IAM users for using OBS. ModelArts data management, development environments, training jobs, and model deployment require OBS for forwarding data.	OBS OperateAccess	Yes
Software Repository for Container (SWR)	Assign permissions to IAM users for using SWR. ModelArts custom images require the SWR FullAccess permission.	SWR OperateAccess	Yes

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Target Service	Description	IAM Permission	Mandatory
Key Management Service (KMS)	To use remote SSH of ModelArts notebook, IAM users require KMS authorization.	DEW KeypairFullAccess KMS CMKFullAccess	No
Intelligent EdgeFabric (IEF)	Assign permissions to IAM users for using IEF. Tenant administrator permissions are required so that ModelArts edge services depending on IEF can be used.	Tenant Administrator	No
Cloud Eye	Assign permissions to IAM users for using Cloud Eye. Using Cloud Eye, you can view the running statuses of ModelArts real-time services and AI application loads, and set monitoring alarms.	CES FullAccess	No
Simple Message Notification (SMN)	Assign permissions to IAM users for using SMN. SMN is used with Cloud Eye.	SMN FullAccess	No
Virtual Private Cloud (VPC)	During the creation of a dedicated resource pool for ModelArts, IAM users require VPC permissions so that they can customize networks.	VPC FullAccess	No
SFS	Assign permissions to IAM users for using SFS. SFS file systems can be mounted to ModelArts dedicated resource pools to serve as storage for development environments or training.	SFS Turbo FullAccess SFS FullAccess	No

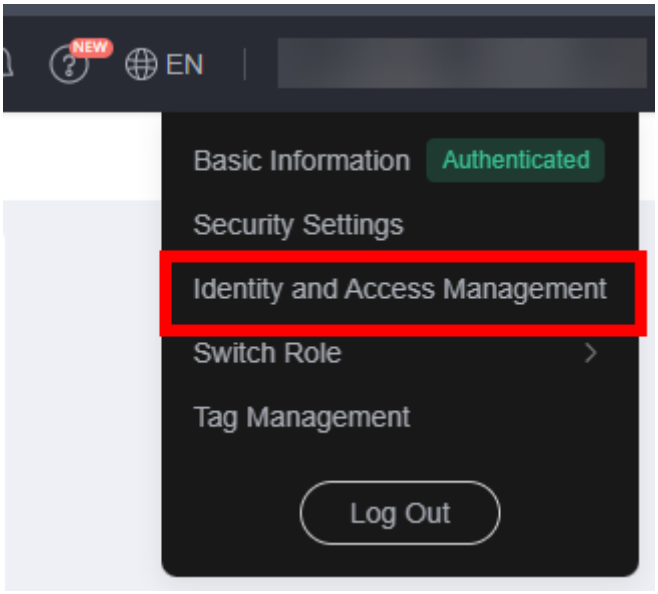
 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

10.6.1.2 Step 1 Creating a User Group and Adding Users to the User Group

Multiple IAM users can be created under a tenant user, and the permissions of the IAM users are managed by group. This section describes how to create a user group and IAM users and add the IAM users to the user group.

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

   Figure 10-16 Identity and Access Management

2. Create a user group. In the navigation pane on the left, choose **User Groups**. Click **Create User Group** in the upper right corner. Then, set **Name** to **UserGroup-2** and click **OK**.

   After the user group is created, the system automatically switches to the user group list. Then, you can add existing IAM users to the user group through user group management. If there is no existing IAM user, create users and add them to the user group.

3. Create IAM users and add them to the user group. In the navigation pane on the left, choose **Users**. On the displayed page, click **Create User** in the upper right corner. On the **Create User** page, add multiple users.

   Set parameters as prompted and click **Next**.

4. On the **Add User to Group** page, select **UserGroup-2** and click **Create**.

   The system will automatically add the two users to the target group one by one.

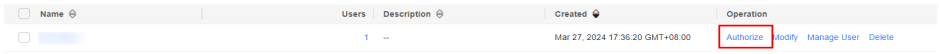
10.6.1.3 Step 2 Assigning Permissions for Using Cloud Services

An IAM user can use cloud services such as ModelArts and OBS only after they are assigned with permissions from the tenant user. This section describes how to assign the permissions to use cloud services to all IAM users in a user group.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

1. On the user group list page of IAM, click **Authorize** of the target user group. The **Authorize User Group** page is displayed.

   Figure 10-17 Authorize

2. Before assigning permissions, learn about minimum permissions requirements of each ModelArts module, as shown in **Table 10-28**.
3. Assign permissions for using ModelArts. Search for ModelArts in the search box. Select either **ModelArts FullAccess** or **ModelArts CommonOperations**.

   The differences between the options are as follows:

   - The users with the ModelArts CommonOperations permission can only use resources, but cannot create, update, or delete any dedicated resource pool. You are advised to assign this permission to IAM users.
   - The users with the ModelArts FullAccess permission have all access permissions, including creating, updating, and deleting dedicated resource pools. Exercise caution when selecting this option.
4. Assign permissions for using OBS. Search for **OBS** and select **OBS Administrator**. ModelArts training jobs use OBS to forward data. Therefore, the permissions for using OBS are necessary.
5. Assign permissions for using SWR. Search for **SWR** and select **SWR FullAccess**. ModelArts custom images require the SWR FullAccess permission.
6. (Optional) Assign the key management permission. Remote SSH of ModelArts notebook requires the key management permission. Search for **DEW** and select **DEW KeypairFullAccess**.

   DEW key management permission is configured in the following regions: **CN North-Beijing1**, **CN North-Beijing4**, **CN East-Shanghai1**, **CN East-Shanghai2**, **CN South-Guangzhou**, **CN Southwest-Guiyang1**, **CN-Hong Kong**, and **AP-Singapore**. In other regions, the KMS key management permission is configured. In this example, the **CN-Hong Kong** region is used. Therefore, the DEW key management permission is to be configured.
7. (Optional) Assign permissions for using IEF. ModelArts requires the Tenant Administrator permission so that edge services depending on IEF can be used.

   Tenant Administrator has the permission to manage all cloud services, not only ModelArts. Exercise caution when assigning the Tenant Administrator permission.
8. (Optional) Assign permissions for using Cloud Eye and SMN. On the details page of a ModelArts real-time service deployed for inference, the number of calls is available. Click **View Details** to obtain more information. If you want to view the overall running status of ModelArts real-time services and AI application loads on Cloud Eye, assign Cloud Eye permissions to IAM users.

   To view monitoring data only, select **CES ReadOnlyAccess**.

   To set alarm monitoring on Cloud Eye, you also need to add **CES FullAccess** and SMN permissions.
9. (Optional) Assign permissions for using VPC. To enable custom network configuration when creating a dedicated resource pool, assign permissions for using VPC.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

10. (Optional) Assign permissions for using SFS and SFS Turbo. To mount an SFS system to a dedicated resource pool as the storage for the development environment or training, assign the permission to use the SFS system.
11. Click **View Selected** in the upper left corner and confirm the selected permissions.
12. Click **Next** and set the minimum authorization scope. Select **Region-specific projects**, select the region to be authorized, and click **OK**.
13. A message is displayed, indicating that the authorization is successful. View the authorization information and click **Finish**. It takes 15 to 30 minutes for the authorization to take effect.

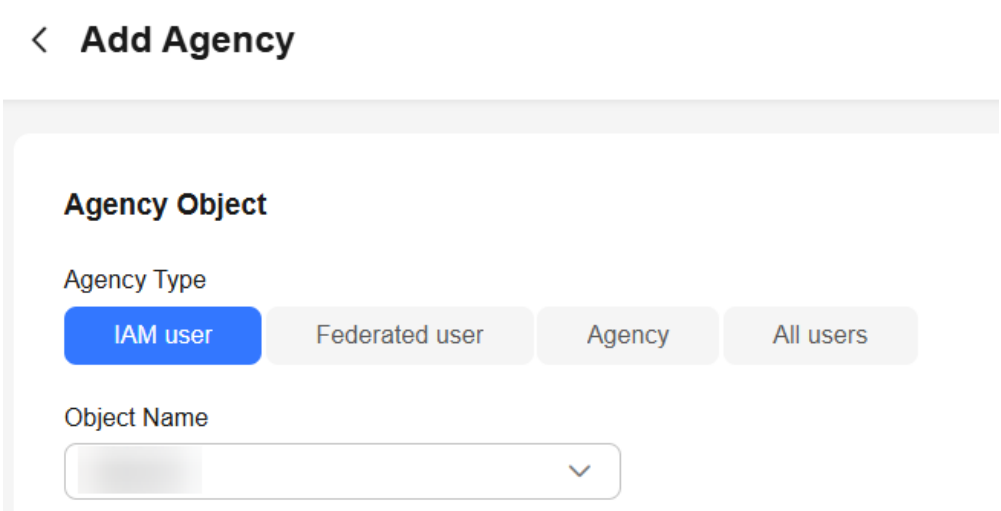
10.6.1.4 Step 3 Configuring Agent-based ModelArts Access Authorization

After assigning IAM permissions, configure ModelArts access authorization for IAM users on ModelArts so that ModelArts can access dependency services such as OBS, SWR, and IEF. In agent-based ModelArts access, only tenant users can set up IAM user permissions. However, IAM users with admin permission can also set up agency authorization.

In this example, the administrator account needs to configure access authorization for IAM users.

1. Log in to the **ModelArts console** as the tenant user. Select your region in the upper left corner.
2. In the navigation pane on the left, click **Permissions** to go to the permission management page.
3. Click **Add Agency**. On the displayed page, set **Agency Type** to **IAM user**, **Object Name** to the target IAM user, and **Agency** to **Add agency**.
4. In the **Permissions** area, set **Permission Template** as required. You can configure the minimum authorization for the IAM user based on the preset permission template.

Figure 10-18 Configuring agency permissions



5. Click **Create** and confirm the information.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

10.6.1.5 Step 4 Verifying User Permissions

It takes 15 to 30 minutes for the permissions configured in 4 to take effect. Therefore, wait for 30 minutes after the configuration and then verify the configuration.

1. Log in to the ModelArts management console as an IAM in **UserGroup-2**. On the login page, ensure that **IAM User Login** is selected.

   Change the password as prompted upon the first login.

2. Check ModelArts permissions.

   a. Select the target region in the upper left corner, which must be the same as that in the authorization configuration.

   b. In the navigation pane on the left of the ModelArts management console, choose **DevEnviron** > **Notebook**. The ModelArts permissions and agency authorization are configured correctly if no message shows insufficient permissions.

      If a message is displayed, indicating that you need to authorize access, the ModelArts agency authorization has not been configured. In this case, follow the instructions in **Step 3 Configuring Agent-based ModelArts Access Authorization** to configure the authorization.

   c. In the navigation pane on the left of the ModelArts management console, choose **DevEnviron** > **Notebook** and click **Create**. If this operation is successful, you have obtained ModelArts operation permissions.

      Alternatively, you can try other functions, such as **Training Management** > **Training Jobs**. If the operation is successful, you can use ModelArts properly.

3. Verify OBS permissions.

   a. In the service list in the upper left corner, select OBS. The OBS management console is displayed.

   b. Click **Create Bucket** in the upper right corner. If this operation is successful, you have obtained OBS operation permissions.

4. Verify SWR permissions.

   a. In the service list in the upper left corner, select SWR. The SWR management console is displayed.

   b. If an SWR page can be properly displayed, you have obtained SWR operation permissions.

5. Verify other optional permissions.
6. Experience ModelArts.

10.6.2 Assigning Basic Permissions for Using Development Environments to an IAM User

Description

This section describes the basic permissions required by IAM users for using development environments. You can assign permissions for specific scenarios by referring to **Permissions List**. In this example, an IAM user is authorized to use a

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

notebook instance for debugging, and data and code are stored in a parallel file system. Use a management account to perform the following configuration operations.

Permissions List

- Permission

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Table 10-29 Permissions required by development environments

Application Scenario	Dependent Service	Dependent Policy	Supported Function	Configuration Suggestion
Lifecycle management of a development environment instance	ModelArts	modelarts:notebook:create modelarts:notebook:list modelarts:notebook:get modelarts:notebook:update modelarts:notebook:delete modelarts:notebook:start modelarts:notebook:stop modelarts:notebook:updateStopPolicy modelarts:image:delete modelarts:image:list modelarts:image:create modelarts:image:get modelarts:pool:list modelarts:tag:list modelarts:network:get aom:metric:get aom:metric:list aom:alarm:list	Start, stop, create, delete, and update an instance. After the **modelarts:notebook:list** or **modelarts:notebook:get** permission is configured, you can open a notebook instance by default.	Recommended. Explicit authorization for these permissions is required only after **strict authorization** is enabled. By default, a newly created account uses the strict authorization mode.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Application Scenario	Dependent Service	Dependent Policy	Supported Function	Configuration Suggestion
Dynamically mounting storage	ModelArts	modelarts:notebook:listMountedStorages modelarts:notebook:mountStorage modelarts:notebook:getMountedStorage modelarts:notebook:umountStorage	Dynamically mount storage.	Configured as required.
	OBS	obs:bucket:ListAllMyBuckets obs:bucket:ListBucket		
Image management	ModelArts	modelarts:image:register modelarts:image:listGroup	Register and view an image on the **Image Management** page.	Configured as required.
Saving an image	SWR	SWR Admin	The **SWR Admin** policy contains the maximum scope of SWR permissions, which can be used to: • Save a running development environment instance as an image. • Create a notebook instance using a custom image.	Configured as required.
Using the SSH function	ECS	ecs:serverKeypairs:list ecs:serverKeypairs:get ecs:serverKeypairs:delete ecs:serverKeypairs:create	Configure a login key for a notebook instance.	Configured as required.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Application Scenario	Dependent Service	Dependent Policy	Supported Function	Configuration Suggestion
	DEW	kps:domainKeypairs:get kps:domainKeypairs:list kps:domainKeypairs:createkmskey		
	KMS	kms:cmk:list		
Mounting an SFS Turbo file system	SFS Turbo	SFS Turbo FullAccess	Read and write an SFS directory as an IAM user. Mount an SFS file system that is not created by you to a notebook instance using a dedicated resource pool as an IAM user.	Configured as required.
Viewing all instances	ModelArts	modelarts:notebook:listAllNotebooks	View development environment instances of all users on the ModelArts management console. The policies are required by the development environment instance administrator.	Configured as required.
	IAM	iam:users:listUsers		

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Application Scenario	Dependent Service	Dependent Policy	Supported Function	Configuration Suggestion
Local VS Code plug-in or PyCharm Toolkit	ModelArts	modelarts:notebook:listAllNotebooks modelarts:trainJob:create modelarts:trainJob:list modelarts:trainJob:update modelarts:trainJobVersion:delete modelarts:trainJob:get modelarts:trainJob:logExport modelarts:workspace:getQuotas (This policy is required if the **workspace** function is enabled.)	Access a notebook instance from local VS Code and submit training jobs.	Configured as required.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Application Scenario	Dependent Service	Dependent Policy	Supported Function	Configuration Suggestion
	OBS	obs:bucket:ListAllMybuckets obs:bucket:HeadBucket obs:bucket:ListBucket obs:bucket:GetBucketLocation obs:object:GetObject obs:object:GetObjectVersion obs:object:PutObject obs:object:DeleteObject obs:object:DeleteObjectVersion obs:object:ListMultipartUploadParts obs:object:AbortMultipartUpload obs:object:GetObjectAcl obs:object:GetObjectVersionAcl obs:bucket:PutBucketAcl obs:object:PutObjectAcl obs:object:ModifyObjectMetaData		
	IAM	iam:projects:listProjects	Obtain an IAM project list through local PyCharm for access configurations.	

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Application Scenario	Dependent Service	Dependent Policy	Supported Function	Configuration Suggestion
VPC access	VPC	VPC ReadOnlyAccess	Mount an instance to your VPC for access of multiple network planes.	Configured as required.

NOTE

To define permissions required to access both global and project-level services, enclose the permissions in two separate custom policies for refined authorization.

- Agency

Table 10-30 Agencies required by development environments

Application Scenario	Dependent Service	Agency Authorization	Description	Configuration Suggestion
JupyterLab	OBS	obs:object:DeleteObject obs:object:GetObject obs:object:GetObjectVersion obs:bucket:CreateBucket obs:bucket:ListBucket obs:bucket:ListAllMyBuckets obs:object:PutObject obs:bucket:GetBucketAcl obs:bucket:PutBucketAcl obs:bucket:PutBucketCORS	Use OBS to upload and download data in JupyterLab through ModelArts notebook.	Recommended.
Development environment monitoring	AOM	aom:alarm:put	Call the AOM API to obtain monitoring data and events of notebook instances and display them in ModelArts notebook.	Recommended.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Application Scenario	Dependent Service	Agency Authorization	Description	Configuration Suggestion
VPC access	VPC	vpc:ports:create vpc:ports:get vpc:ports:delete vpc:subnets:get	Mount an instance to your VPC for access of multiple network planes.	Configured as required.

Procedure

In this example, a notebook instance is used to debug code on a single node with a single PU. The data and code are stored in an OBS parallel file system. After the debugging, you can save the notebook instance as an image.

Step 1 Log in to the 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.

Step 2 Add the permissions to use development environments and the permissions of the dependent service SWR. On the IAM console, choose **Permissions** > **Policies/Roles** from the navigation pane, click **Create Custom Policy** in the upper right corner, and configure policies.

1. Add the permissions to use development environments.
   - **Policy Name**: Enter a policy name, for example, **notebook**.
   - **Policy View**: Select **JSON**.
   - **Policy Content**: Enter the following information:

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "modelarts:notebook:create",
                "modelarts:notebook:list",
                "modelarts:notebook:get",
                "modelarts:notebook:update",
                "modelarts:notebook:delete",
                "modelarts:notebook:start",
                "modelarts:notebook:stop",
                "modelarts:notebook:updateStopPolicy",
                "modelarts:notebook:listMountedStorages",
                "modelarts:notebook:mountStorage",
                "modelarts:notebook:getMountedStorage",
                "modelarts:notebook:umountStorage",
                "modelarts:image:delete",
                "modelarts:image:list",
                "modelarts:image:create",
                "modelarts:image:get",
```

```
            "modelarts:pool:list",
            "modelarts:tag:list",
            "modelarts:network:get",
            "aom:metric:get",
            "aom:metric:list",
            "aom:alarm:list"
        ]
    }
  ]
}
```

Step 3 Add the permissions of the dependent service OBS.

On the IAM console, choose **Permissions** > **Policies/Roles** from the navigation pane, click **Create Custom Policy** in the upper right corner, and configure policies.

- **Policy Name**: Enter a policy name, for example, **notebook-obs**.
- **Policy View**: Select **JSON**.
- **Policy Content**: Enter the following information:

```
{
  "Version": "1.1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "obs:bucket:ListAllMyBuckets",
        "obs:bucket:ListBucket"
      ]
    }
  ]
}
```

NOTE

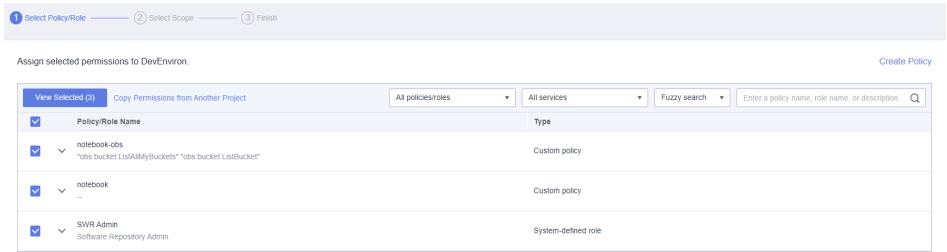
To define permissions required to access both global and project-level services, enclose the permissions in two separate custom policies for refined authorization. **trainJob** is for project-level services, and **trainJobobs** is for global services. **Learn more**.

Step 4 Create a user group and add the required users to the user group. For details, see **Step 1 Creating a User Group and Adding Users to the User Group**.

Step 5 Authorize the policy for the user group.

On the user group list page of IAM, click **Authorize** of the target user group. The **Authorize User Group** page is displayed. Select **notebook**, **notebook-obs**, and **SWR Admin**. Click **Next** and then **OK**.

Figure 10-19 Authorizing a policy to a user group



Step 6 Create a ModelArts agency.

1. Create an agency authorization policy.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

On the IAM console, choose **Permissions** > **Policies/Roles** from the navigation pane, click **Create Custom Policy** in the upper right corner, and configure policies.

- **Policy Name**: Enter a policy name, for example, **ma_agency_obs**.
- **Policy View**: Select **JSON**.
- **Policy Content**: Enter the following information:

```
{
   "Version": "1.1",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "obs:object:GetObject",
            "obs:object:DeleteObject",
            "obs:bucket:PutBucketAcl",
            "obs:object:PutObject",
            "obs:bucket:CreateBucket",
            "obs:bucket:GetBucketAcl",
            "obs:bucket:PutBucketCORS",
            "obs:bucket:ListAllMyBuckets",
            "obs:bucket:ListBucket",
            "obs:object:GetObjectVersion"
         ]
      }
   ]
}
```

2. Create an agency.

   On the IAM console, choose **Agencies** from the navigation pane, click **Create Agency** in the upper right corner, and configure policies. Enter agency information and click **Next**.

   - **Agency Name**: Customize an agency name, for example, **ma_agency_notebook**.
   - **Agency Type**: Select **Cloud service**.
   - **Cloud Service**: Select **ModelArts**.
   - **Validity Period**: Select **Unlimited**.

   Select the new agency policy and click **Next**. Set the minimum authorization scope. Select **All resources** for **Scope**. Click **OK**.

3. Assign ModelArts permissions for IAM users.

   Log in to the ModelArts management console. In the navigation pane on the left, choose **Permission Management**. On the displayed page, click **Add Agency**. Set **Object Name** to the target IAM user and select the created agency. Then, click **Create**.

   NOTE

   Authorization already exists. Updating it will replace the existing one, which may impact your services. This operation cannot be undone.

Step 7 Verify that the permissions are configured.

Log in as the IAM user. If you can **create a notebook instance**, **mount an OBS file system** (the OBS bucket must be created by the administrator), and **save an image** on the console, the agency configuration is successful.

----End

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

10.6.3 Assigning Basic Permissions for Using New-Version Inference to an IAM User

Scenario

This section describes the basic permissions required by sub-accounts in the inference deployment scenario. This example involves granting sub-accounts the permissions to deploy inference services using the real-time service (new version) function on the ModelArts console.

Procedure

In this example, the image is uploaded to SWR, and the inference service is deployed on the real-time service (new version) page of the ModelArts console.

Step 1 Log in to the 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 console.

Step 2 Add permissions for deploying services. On the IAM console, choose **Permissions** > **Policies/Roles** from the navigation pane on the left, click **Create Custom Policy** in the upper right corner, and configure the policies.

Add permissions for deploying services.

- **Policy Name**: Enter a policy name, for example, **service**.
- **Policy View**: Select **JSON**.
- **Policy Content**: Enter the following information.

```
{
  "Version": "1.1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "lts:groups:create",
        "lts:groups:get",
        "lts:groups:list",
        "lts:topics:create",
        "lts:topics:delete",
        "lts:topics:get",
        "lts:topics:list",
        "obs:bucket:GetBucketLocation",
        "obs:bucket:HeadBucket",
        "obs:bucket:ListAllMybuckets",
        "obs:bucket:ListBucket",
        "obs:bucket:PutBucketAcl",
        "obs:object:AbortMultipartUpload",
        "obs:object:DeleteObject",
        "obs:object:DeleteObjectVersion",
        "obs:object:GetObject",
        "obs:object:GetObjectAcl",
        "obs:object:GetObjectVersion",
        "obs:object:GetObjectVersionAcl",
        "obs:object:ListMultipartUploadParts",
        "obs:object:PutObject",
        "obs:object:PutObjectAcl",
        "swr:instance:list",
        "swr:repo:listRepoTags",
        "swr:repository:getTag",
        "swr:repository:listArtifacts",
        "swr:repository:listRepositories",
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
            "csms:secret:list",
            "csms:secretVersion:get",
            "sfsturbo:shares:getAllShares",
            "sfsturbo:shares:getShare",
            "sfsturbo:shares:showFsDir",
            "modelarts:authorization:list",
            "modelarts:clusterFlavor:list",
            "modelarts:image:get",
            "modelarts:image:list",
            "modelarts:image:listGroup",
            "modelarts:network:list",
            "modelarts:pool:get",
            "modelarts:pool:list",
            "modelarts:pool:update",
            "modelarts:service:create",
            "modelarts:service:get",
            "modelarts:service:list",
            "modelarts:tag:listTags",
            "modelarts:warmupTask:list",
            "modelarts:workspace:checkAuthorizations",
            "modelarts:workspace:get",
            "modelarts:workspace:getAuthMode",
            "modelarts:workspace:list",
            "vpc:vpcs:list",
            "vpc:subnets:get",
            "kms:dek:decrypt"
        ]
    }
  ]
}
```

Step 3 Create a user group and add the target user to it. For details, see **Step 1 Creating a User Group and Adding Users to the User Group**.

Step 4 Grant the required policy to the user group.

On the user group list page of IAM, click **Authorize** of the target user group. The **Authorize User Group** page is displayed. Select **service** and **SWR Admin**. Click **Next** and then **OK**.

Step 5 Create a ModelArts agency.

1. Create an agency authorization policy.

   On the IAM console, choose **Permissions** > **Policies/Roles** from the navigation pane on the left, click **Create Custom Policy** in the upper right corner, and configure the policies.

   - **Policy Name**: Enter a policy name, for example, **service_agency**.
   - **Policy View**: Select **JSON**.
   - **Policy Content**: Enter the following information:

     ```
     {
       "Version": "1.1",
       "Statement": [
         {
           "Effect": "Allow",
           "Action": [
             "lts:groups:create",
             "lts:groups:get",
             "lts:groups:list",
             "lts:topics:create",
             "lts:topics:delete",
             "lts:topics:get",
             "lts:topics:list",
             "sfsturbo:shares:getShare",
             "sfsturbo:shares:showFsDir",
             "swr:repo:listRepoTags",
     ```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
            "swr:repository:getTag",
            "obs:object:DeleteObject",
            "obs:object:GetObject",
            "obs:bucket:CreateBucket",
            "obs:bucket:ListBucket",
            "obs:object:PutObject",
            "obs:bucket:GetBucketAcl",
            "obs:bucket:PutBucketAcl",
            "obs:bucket:PutBucketCORS",
            "kms:dek:decrypt"
            "csms:secretVersion:get",
            "modelarts:authorization:list",
            "modelarts:image:get",
            "modelarts:pool:get",
            "modelarts:pool:list",
            "modelarts:pool:update",
            "modelarts:warmupTask:list"
        ]
    }
  ]
}
```

2. Create an agency.

   On the IAM console, choose **Agencies** from the navigation pane, click **Create Agency** in the upper right corner, and configure policies. Enter agency information and click **Next**.

   - **Agency Name**: Enter an agency name, for example, **ma_agency_service**.
   - **Agency Type**: Select **Cloud service**.
   - **Cloud Service**: Select **ModelArts**.
   - **Validity Period**: Select **Unlimited**.

   Select the new agency policy and click **Next**. Set the minimum authorization scope. Select **All resources** for **Scope**. Click **OK**.

3. Assign ModelArts permissions for IAM users.

   Log in to the ModelArts console. In the navigation pane on the left, choose **Permission Management**. On the displayed page, click **Add Authorization**. Set **Authorized User** to **IAM user** and select the created agency. Then, click **Create**.

Step 6 Verify that the permission is assigned.

Log in to the ModelArts console as the IAM user. If the user can upload images to SWR and deploy them, the permission configuration is successful.

----End

10.6.4 Separately Assigning Permissions to Administrators and Developers

In small- and medium-sized teams, administrators need to globally control ModelArts resources, and developers only need to focus on their own instances. Generally, the **te_admin** permission of a developer account must be configured by the tenant account. This section uses notebook as an example to describe how to assign different permissions to administrators and developers through custom policies.

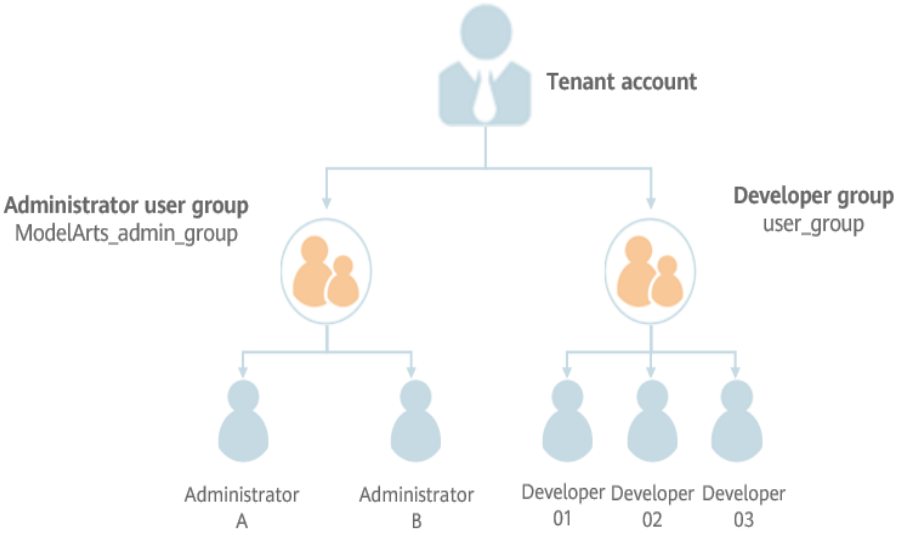
 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Scenarios

To develop a project using notebook, administrators need full control permissions for using ModelArts dedicated resource pools, and access and operation permissions on all notebook instances.

To use development environments, developers only need operation permissions for using their own instances and dependent services. They do not need to perform operations on ModelArts dedicated resource pools or view notebook instances of other users.

Figure 10-20 Account relationships



Configuring Permissions for an Administrator

Assign full control permissions to administrators for using ModelArts dedicated resource pools and all notebook instances. The procedure is as follows:

Step 1 Use a tenant account to create an administrator user group **ModelArts_admin_group** and add administrator accounts to **ModelArts_admin_group**. For details, see **Step 1 Creating a User Group and Adding Users to the User Group**.

Step 2 Create a custom policy.

1. Log in to the **IAM console**.
2. Create custom policy 1 and assign IAM and OBS permissions to the user. In the navigation pane of the IAM console, choose **Permissions** > **Policies/Roles**. Click **Create Custom Policy** in the upper right corner. On the displayed page, enter **Policy1_IAM_OBS** for **Policy Name**, select **JSON** for **Policy View**, configure the policy content, and click **OK**.

   The custom policy **Policy1_IAM_OBS** is as follows, which grants IAM and OBS operation permissions to the user. You can directly copy and paste the content.

```
{
    "Version": "1.1",
    "Statement": [
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
        {
            "Effect": "Allow",
            "Action": [
                "iam:users:listUsers",
                "iam:projects:listProjects",
                "obs:object:PutObject",
                "obs:object:GetObject",
                "obs:object:GetObjectVersion",
                "obs:bucket:HeadBucket",
                "obs:object:DeleteObject",
                "obs:bucket:CreateBucket",
                "obs:bucket:ListBucket"
                ]
        }
    ]
}
```

3. Repeat **Step 2.2** to create custom policy 2 and grant the user the permissions to perform operations on dependent services ECS, SWR, MRS, and SMN as well as ModelArts. Set **Policy Name** to **Policy2_AllowOperation** and **Policy View** to **JSON**, configure the policy content, and click **OK**.

   The custom policy **Policy2_AllowOperation** is as follows, which grants the user the permissions to perform operations on dependent services ECS, SWR, MRS, and SMN as well as ModelArts. You can directly copy and paste the content.

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:serverKeypairs:list",
                "ecs:serverKeypairs:get",
                "ecs:serverKeypairs:delete",
                "ecs:serverKeypairs:create",
                "swr:repository:getNamespace",
                "swr:repository:listNamespaces",
                "swr:repository:deleteTag",
                "swr:repository:getRepository",
                "swr:repository:listTags",
                "swr:instance:createTempCredential",
                "mrs:cluster:get",
                "modelarts:*:*"
            ]
        }
    ]
}
```

Step 3 Grant the policy created in **Step 2** to the administrator group **ModelArts_admin_group**.

1. In the navigation pane of the IAM console, choose **User Groups**. On the **User Groups** page, locate the row that contains **ModelArts_admin_group**, click **Authorize** in the **Operation** column, and select **Policy1_IAM_OBS** and **Policy2_AllowOperation**. Click **Next**.
2. Specify the scope as **All resources** and click **OK**.

Step 4 Configure ModelArts agency authorization for the administrator to use the development environment and create dedicated resource pools.

1. Log in to the **ModelArts console**. In the navigation pane on the left, choose **Permission Management**.
2. Click **Add Agency**. On the **Add Agency** page, select **IAM user** for **Agency Type**, select the administrator account for **Object Name**, select **Add agency**

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

for **Agency**, and select **Development Environment** and **Dedicated Resource Pool** for **Agency Template**.

3. Click **Create**.

Step 5 Test administrator permissions.

1. Log in to the **ModelArts console** as an administrator. On the login page, ensure that **IAM User Login** is selected.

   Change the password as prompted upon the first login.

2. In the navigation pane of the ModelArts management console, choose **Dedicated Resource Pools** and click **Create**. If the console does not display a message indicating insufficient permissions, the permissions have been assigned to the administrator.

----End

Configuring Permissions for a Developer

Use IAM for fine-grained control of developer permissions. The procedure is as follows:

Step 1 Use a tenant account to create a developer user group **user_group** and add developer accounts to **user_group**. For details, see **Step 1 Creating a User Group and Adding Users to the User Group**.

Step 2 Create a custom policy.

1. Log in to the **IAM console**.
2. Create custom policy 3 to prevent users from performing operations on ModelArts dedicated resource pools and viewing notebook instances of other users.

   In the navigation pane of the IAM console, choose **Permissions** > **Policies/Roles**. Click **Create Custom Policy** in the upper right corner. On the displayed page, enter **Policy3_DenyOperation** for **Policy Name**, select **JSON** for **Policy View**, configure the policy content, and click **OK**.

   The custom policy **Policy3_DenyOperation** is as follows. You can copy and paste the content.

```
{
   "Version": "1.1",
   "Statement": [
         {
          "Effect": "deny",
          "Action": [
              "modelarts:pool:create",
              "modelarts:pool:update",
              "modelarts:pool:delete",
      "modelarts:notebook:listAllNotebooks"
            ]

         }
   ]
}
```

Step 3 Grant the custom policy to the developer user group **user_group**.

1. In the navigation pane of the IAM console, choose **User Groups**. On the **User Groups** page, locate the row that contains **user_group**, click **Authorize** in the **Operation** column, and select **Policy1_IAM_OBS**, **Policy2_AllowOperation**, and **Policy3_DenyOperation**. Click **Next**.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

2. Specify the scope as **All resources** and click **OK**.

Step 4 Configure ModelArts agency authorization for developers to use only the development environment.

1. Log in to the **ModelArts console**. In the navigation pane on the left, choose **Permission Management**.
2. Click **Add Agency**. On the **Add Agency** page, select **IAM user** for **Agency Type**, select the developer account for **Object Name**, select **Add agency** for **Agency**, set **Agency Name** to **ma_agency_develop_user**, and select **Development Environment** for **Agency Template**.
3. Click **Create**.
4. On the **Permission Management** page, click **Add Agency** again. On the **Add Agency** page that is displayed, configure an agency for other developer users.

   Set **Agency Type** to **IAM user**, select a developer account for **Object Name**, and select the existing agency **ma_agency_develop_user** created before.

Step 5 Test developer permissions.

1. Log in to the **ModelArts console** as an IAM user in **user_group**. On the login page, ensure that **IAM User Login** is selected.

   Change the password as prompted upon the first login.
2. In the navigation pane of the ModelArts management console, choose **Dedicated Resource Pools** and click **Create**. If the console displays a message indicating insufficient permissions, the permissions have been assigned to the developer.

----End

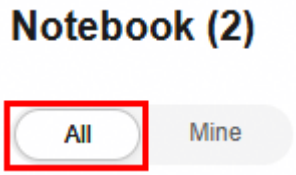
10.6.5 Assigning the Required Permissions

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.

- Viewing all notebook instances: **Grant the permission to the IAM user for viewing all notebook instances**. Go to the **Notebook** page. On the displayed page, click **All**.

  Figure 10-21 Viewing all

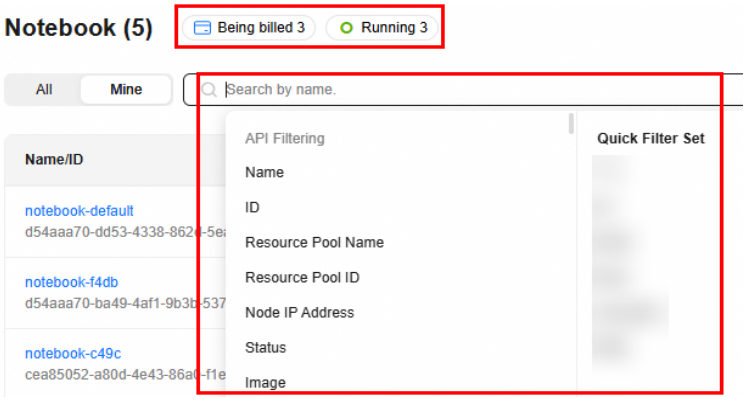
  

- Filtering notebook instances:
  - In the upper part of the page, click **Being billed** or **Running** to filter notebook instances. Hover over **Being billed** to view the number of notebook instances currently consuming billing resources (compute and storage).

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

- The search box defaults to **Name**. You can also filter by **ID**, **Resource Pool Name**, **Resource Pool ID**, **Node IP Address**, or **Resource Tag** as needed.

Figure 10-22 Filtering notebook instances



After selecting filter criteria or entering content in the search box, click on the right to save these criteria as a quick filter set for later use.

Figure 10-23 Saving as a quick filter set



Assigning the Permissions to Manage All Notebook Instances to an IAM User

Any IAM user with **modelarts:notebook:listAllNotebooks** and **iam:users:listUsers** permissions can click **View all** on the notebook page to view and open all instances in the current IAM project. After the permission is granted, you can access OBS and SWR of IAM users in a notebook instance.

CAUTION

Once the **modelarts:notebook:listAllNotebooks** permission is given to an IAM user, they can view and manage notebook instances created by others. This includes actions like opening, starting, stopping, and deleting. Exercise caution when granting this permission.

1. Log in to the **ModelArts console** 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 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.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

- **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.

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

To remotely access another user's notebook instance via SSH after viewing all notebook instances, update the SSH key pair with your own key. Otherwise, errors like "ModelArts.6789=The current user does not have the permission to use the SSH key pair {0}, please update the key pair of the instance and try again." or "ModelArts.6863=Specified ssh keypair {0} cannot be retrieved on ECS keypair management, please check the ECS keypair access." may occur. For details about how to update a key pair, see **Enabling Remote SSH**.

10.6.6 Logging In to a Training Container Using Cloud Shell

Application Scenario

You can use Cloud Shell provided by the ModelArts console to log in to a running training container.

Constraints

Only dedicated resource pools support Cloud Shell. The training job must be in the **Running** state.

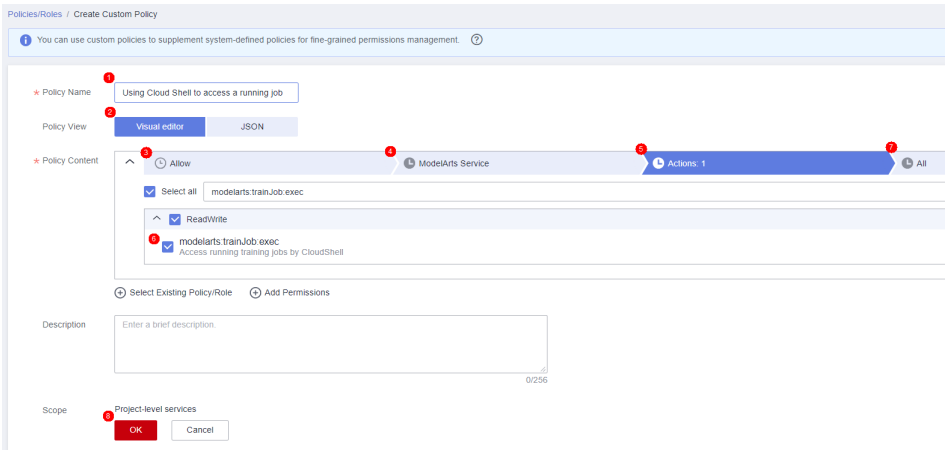
Preparation: Assigning the Cloud Shell Permission to an IAM User

1. Log in to the Huawei Cloud 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.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

2. On the IAM console, choose **Permissions** > **Policies/Roles** from the navigation pane, click **Create Custom Policy** in the upper right corner, and configure the following parameters.
   - **Policy Name**: Enter a custom policy name, for example, **Using Cloud Shell to access a running job**.
   - **Policy View**: Select **Visual editor**.
   - **Policy Content**: Select **Allow**, **ModelArts Service**, **modelarts:trainJob:exec**, and default resources.

   Figure 10-24 Creating a custom policy



3. In the navigation pane, choose **User Groups**. Then, click **Authorize** in the **Operation** column of the target user group. On the **Authorize User Group** page, select the custom policies created in 2, and click **Next**. Then, select the scope and click **OK**.

   After the configuration, all users in the user group have the permission to use Cloud Shell to log in to a running training container.

   If no user group is available, create a user group, add users using the user group management function, and configure authorization. If the target user is not in a user group, you can add the user to a user group through the user group management function.

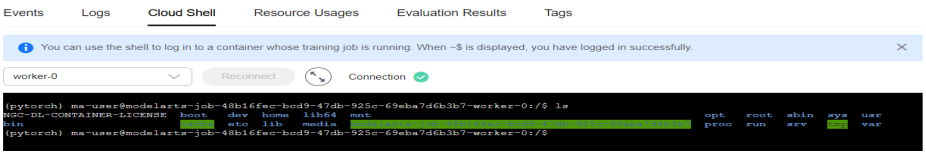
Using Cloud Shell

1. Configure parameters based on **Preparation: Assigning the Cloud Shell Permission to an IAM User**.
2. On the ModelArts console, choose **Model Training** > **Training Jobs** from the navigation pane.
3. In the training job list, click the name of the target job to go to the training job details page.
4. On the training job details page, click the **Cloud Shell** tab and log in to the training container.

   Verify that the login is successful, as shown in the following figure.

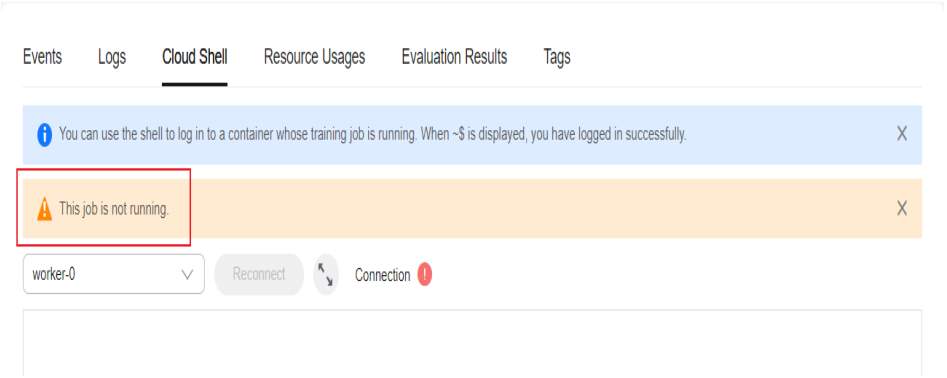
 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Figure 10-25 Cloud Shell page



If the job is not running or the permission is insufficient, Cloud Shell cannot be used. In this case, locate the fault as prompted.

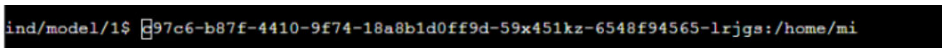
Figure 10-26 Error message



NOTE

A path display exception may occur when you log in to the Cloud Shell page. In this case, press **Enter** to rectify the fault.

Figure 10-27 Abnormal path



10.6.7 Prohibiting a User from Using a Public Resource Pool

This section describes how to control the ModelArts permissions of a user so that the user is not allowed to use a public resource pool to create training jobs, create notebook instances, or deploy inference services.

Context

Through permission control, ModelArts dedicated resource pool users can be prohibited from using a public resource pool to create training jobs, create notebook instances, or deploy inference services.

To control the permissions, configure the following permission policy items:

- **modelarts:notebook:create**: allows you to create a notebook instance.
- **modelarts:trainJob:create**: allows you to create a training job.
- **modelarts:service:create**: allows you to create an inference service.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Procedure

1. Log in to the 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. In the navigation pane, choose **Permissions** > **Policies/Roles**. On the **Policies/Roles** page, click **Create Custom Policy** in the upper right corner, configure parameters, and click **OK**.
   - **Policy Name**: Configure the policy name.
   - **Policy View**: Select **Visual editor** or **JSON**.
   - **Policy Content**: Select **Deny**. In **Select service**, search for **ModelArts** and select it. In **ReadWrite** under **Actions**, search for **modelarts:trainJob:create**, **modelarts:notebook:create**, and **modelarts:service:create** and select them. **All**: Retain the default setting. In **Add request condition**, click **Add Request Condition**. In the displayed dialog box, set **Condition Key** to **modelarts:poolType**, **Operator** to **StringEquals**, and **Value** to **public**.

     The policy content in JSON view is as follows:

     ```
     {
        "Version": "1.1",
        "Statement": [
           {
              "Effect": "Deny",
              "Action": [
                 "modelarts:trainJob:create",
                 "modelarts:notebook:create",
                 "modelarts:service:create"
              ],
              "Condition": {
                 "StringEquals": {
                    "modelarts:poolType": [
                       "public"
                    ]
                 }
              }
           }
        ]
     }
     ```
3. In the navigation pane, choose **User Groups**. On the **User Groups** page, locate the row containing the target user group and click **Authorize** in the **Operation** column. On the **Authorize User Group** page, select the custom policy 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 one, add users to it through user group management, and configure authorization for the user group. If the target user is not in a user group, add the user to a user group through user group management.
4. Add the policy to the user's agency authorization. This prevents the user from breaking the permission scope through a token on the tenant plane.

   In the navigation pane, choose **Agencies**. Locate the agency used by the user group on ModelArts and click **Modify** in the **Operation** column. In the **Permissions** tab, click **Authorize**, select the created custom policy, and click **Next**. Select the scope for authorization and click **OK**.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Verification

Log in to the ModelArts console as an IAM user, choose **Model Training** > **Training Jobs**, and click **Create Training Job**. On the page for creating a training job, only a dedicated resource pool can be selected for **Resource Pool**.

Log in to the ModelArts console as an IAM user, choose **Development Workspace** > **Notebook**, and click **Create**. On the page for creating a notebook instance, only a dedicated resource pool can be selected for **Resource Pool**.

Log in to the ModelArts console as an IAM user, choose **Model Deployment** > **Real-Time Services**, and click **Deploy**. On the page for service deployment, only a dedicated resource pool can be selected for **Resource Pool**.

10.6.8 Authorizing ModelArts to Use SFS Turbo

This section describes how to configure ModelArts agency permissions to enable you to associate and disassociate your dedicated resource pool network with an SFS Turbo file system.

- To add an agency and authorize it to perform operations on SFS Turbo, see **Adding an Agency and Granting Permissions for SFS Turbo**.
- To add permissions to an existing agency for SFS Turbo, see **Adding Permissions to an Existing Agency for SFS Turbo**.

Context

After creating a network on the console, you can click **More** in the **Operation** column and choose **Add SFS Turbo** or **Disassociate** on the resource pool network list page. When you associate a network with an SFS Turbo file system, the file system joins the network and becomes available for training and development purposes.

Before you can associate or disassociate an SFS Turbo file system, you must grant ModelArts the permissions to perform operations on SFS Turbo.

To control the permissions, configure the following permission policy items:

- sfsturbo:shares:addShareNic: allows you to add NICs.
- sfsturbo:shares:deleteShareNic: allows you to delete NICs.
- sfsturbo:shares:showShareNic: allows you to obtain NIC details.
- sfsturbo:shares:listShareNics: allows you to obtain NICs.

Constraints

This function is available in certain regions.

Adding an Agency and Granting Permissions for SFS Turbo

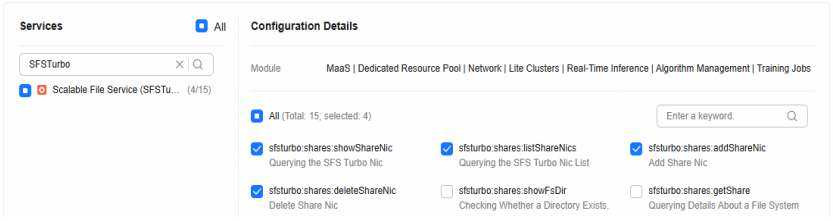
1. Log in to the **ModelArts console**. In the navigation pane on the left, choose **Permission Management**.
2. Click **Add Agency**. On the displayed page, configure parameters.
   - **Agency Type**: Select **IAM user**, **Federated user**, **Agency**, or **All users** as required.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

- **Object Name**: Select an object name. Ensure that the user group of this object has completed IAM authorization. Otherwise, subsequent agency configurations cannot take effect.
- **Agency**: Select **Add agency**.
- **Agency Configuration**: Select **Common**. In the service list, select **sfsturbo:shares:addShareNic**, **sfsturbo:shares:deleteShareNic**, **sfsturbo:shares:showShareNic**, and **sfsturbo:shares:listShareNics** under SFS Turbo.

3. Click **Create**. After you authorize the agency, any user with the permission can associate or disassociate SFS Turbo.

Figure 10-28 Creating an agency and granting certain permissions for ModelArts

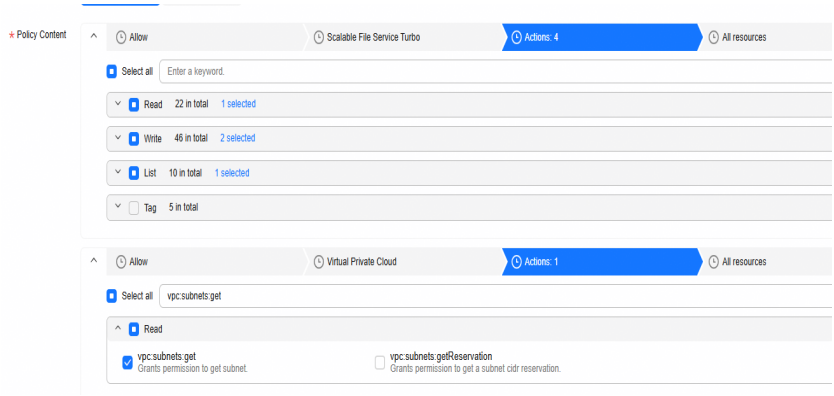


Adding Permissions to an Existing Agency for SFS Turbo

1. Log in to the 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 console of the new version.
2. In the navigation pane of the IAM console, choose **Identity Policies**. Click **Create Custom Policy** in the upper right corner, configure parameters, and click **OK**.
   - Policy Name: Enter a custom policy name, for example, **Authorizing ModelArts to use SFS Turbo**.
   - **Policy View**: Select **Visual editor** or **JSON**.
   - **Policy Content**: Select **Allow**. In **Select service**, search for **SFSTurbo** and select it. In **ReadOnly** under **Actions**, search for **sfsturbo:shares:showShareNic** and **sfsturbo:shares:listShareNics** and select them. In **ReadWrite** under **Actions**, search for **sfsturbo:shares:addShareNic** and **sfsturbo:shares:deleteShareNic** and select them. Select **All** for **All**.
   - The sfsturbo:shares:addShareNic action depends on the vpc:subnets:get permission. If the permissions are not granted, click on the right to grant them.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Figure 10-29 Policy content



The policy content in JSON view is as follows:

```
{
  "Version": "5.0",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "vpc:subnets:get"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "sfsturbo:shares:addShareNic",
        "sfsturbo:shares:deleteShareNic",
        "sfsturbo:shares:listShareNics",
        "sfsturbo:shares:showShareNic"
      ]
    }
  ]
}
```

3. The **modelarts_agency** agency is used as an example. On the IAM console, click **Agencies**. Locate **modelarts_agency** and click **Authorize** in the **Operation** column. Choose the custom policy created in **2** and click **Next**. Then, specify the authorization scope and click **OK**.

   After you authorize the agency, any user with the permission can associate or disassociate SFS Turbo.

Verification

Log in to the **ModelArts console**. In the navigation pane on the left, choose **Resource Management** > **Network**. Locate the target network and choose **More** > **Add SFS Turbo** in the **Operation** column.

Log in to the **ModelArts console**. In the navigation pane on the left, choose **Resource Management** > **Network**. Locate the target network and choose **More** > **Disassociate SFS Turbo** in the **Operation** column.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

10.6.9 Assigning SFS Turbo Folder-Level Access Permissions to an IAM User

Scenario

Grant access permission of specific SFS Turbo folders to IAM users.

NOTE

Granting the IAM user the SFS Turbo folder-level access permission is a whitelist function. Submit a service ticket to apply for the permission as needed.

Constraints

- Ensure that you have enabled strict authorization. By default, a newly created account uses the strict authorization mode. The strict authorization mode switch is displayed on the **Permission Management** page of the old ModelArts console. The new console does not provide the strict authorization switch.
- If the ModelArts permission for the IAM user were not configured prior to enabling strict authorization mode, the IAM user might lose access to ModelArts once that mode is activated. Configure ModelArts permissions based on the service requirements. For details, see **Dependencies and Agencies**.

Procedure

Step 1 Log in to the management console using the main account, 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.

Step 2 On the IAM console, choose **Permissions** > **Policies/Roles** from the navigation pane on the left, click **Create Custom Policy** in the upper right corner, and configure the policy as follows:

- **Policy Name**: Enter a policy name, for example, **ma_sfs_turbo**.
- **Policy View**: Select **JSON**.
- **Policy Content**: Enter the following information:

```
{
   "Version": "1.1",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "<modelarts_action>"
         ],
         "Condition": {
            "StringEqualsIfExists": {
               "modelarts:sfsId": [
                  "<your_sfs_id>"
               ],
               "modelarts:sfsPath": [
                  "<sfs_path>"
               ],
               "modelarts:sfsOption": [
                  "<sfs_option>"
               ]
            }
```

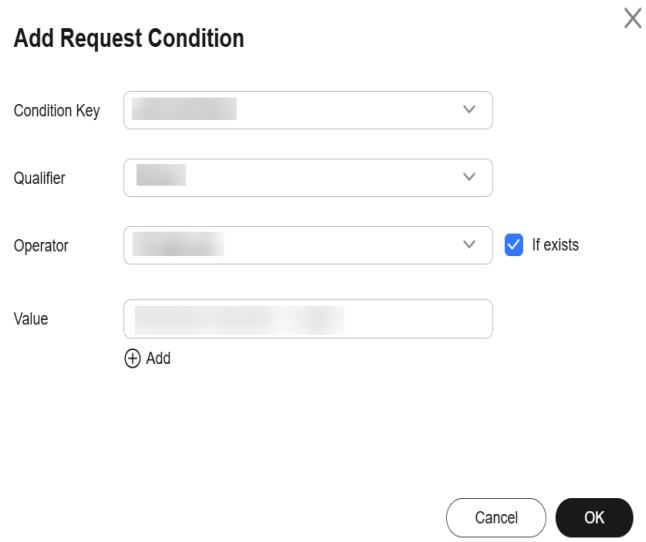
 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
        }
      }
    ]
}
```

NOTE

- Before the preceding policies are created, all IAM users can mount to SFS Turbo by default. After you create the preceding SFS permission control policies, the IAM users who are not granted with the permission cannot mount to SFS Turbo when creating a training job on the ModelArts console, except the IAM users with the **Tenant Administrator** permission.
- Currently, you can configure only permit policies, that is, the **Effect** in the policy can only be set to **Allow**. Do not configure deny policies.
- The **Condition** parameter must use the **StringEqualsIfExists** field, and **If exists** must be enabled for the corresponding visualized view.

Figure 10-30 Enabling **If exists**



Replace *<modelarts_action>*, *<your_sfs_id>*, *<sfs_path>*, and *<sfs_option>* with actual parameters as you need. The following table describes the parameters.

Table 10-31 Parameter description

Parameter	Description

Action	Scenario in which the SFS Turbo folder access permission is granted.
	• **modelarts:notebook:create**: The permission is granted during development environment instance creation.
	• **modelarts:trainJob:create**: The permission is granted during training job creation.
	Multiple actions are supported, the following shows an example:
	`"Action": [`
	`  "modelarts:trainJob:create",`
	`  "modelarts:notebook:create"`
	`],`

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Parameter	Description
modelarts:sfsId	SFS Turbo ID, which can be obtained on the SFS Turbo details page. You can enter multiple IDs, the following shows an example: "modelarts:sfsId": ["0e51c7d5-d90e-475a-b5d0-ecf896da3b0d", "2a70da1e-ea87-4ee4-ae1e-55df846e7f41"],
modelarts:sfsPath	Path of the SFS Turbo folder whose permissions need to be configured. You can enter multiple paths, the following shows an example: "modelarts:sfsPath": ["/path1", "/path2/path2-1"], If there are multiple SFS IDs, the SFS paths will apply to all SFS IDs. As shown in the following example, permission to access **/path1** and **/path2/path2-1** of both **0e51c7d5-d90e-475a-b5d0-ecf896da3b0d** and **2a70da1e-ea87-4ee4-ae1e-55df846e7f41** are configured. "modelarts:sfsId": ["0e51c7d5-d90e-475a-b5d0-ecf896da3b0d", "2a70da1e-ea87-4ee4-ae1e-55df846e7f41"], "modelarts:sfsPath": ["/path1", "/path2/path2-1"],

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Parameter	Description

modelarts:sfsOption	Type of the access permission. The following parameters are supported:
	• **readonly**: Read-only permission
	• **readwrite**: Read and write permission. When you create a development environment instance, **modelarts:notebook:create** must be set to **readwrite**.
	To add multiple SFS options to a custom policy, add a JSON structure to Statement, the following shows an example:
	{
	"Version": "1.1",
	"Statement": [
	{
	"Effect": "Allow",
	"Action": [
	"modelarts:trainJob:create"
	],
	"Condition": {
	"StringEqualsIfExists": {
	"modelarts:sfsId": [
	"0e51c7d5-d90e-475a-b5d0-ecf896da3b0d"
	],
	"modelarts:sfsPath": [
	"/path1"
	],
	"modelarts:sfsOption": [
	"readonly"
	]
	}
	}
	},
	{
	"Effect": "Allow",
	"Action": [
	"modelarts:trainJob:create"
	],
	"Condition": {
	"StringEqualsIfExists": {
	"modelarts:sfsId": [
	"0e51c7d5-d90e-475a-b5d0-ecf896da3b0d"
	],
	"modelarts:sfsPath": [
	"/path2"
	],
	"modelarts:sfsOption": [
	"readwrite"
	]
	}
	}
	}
	]
	}

Step 3 Create a user group and add the user to the user group. For details, see **Step 1 Creating a User Group and Adding Users to the User Group**.

Step 4 Grant a policy to the user group. On the user group list page of IAM, click **Authorize** of the target user group. The **Authorize User Group** page is displayed. Select the **ma_sfs_turbo** policy created in **Step 2**. Click **Next** and then **OK**.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Step 5 Add the **IAM ReadOnlyAccess** permission to an existing ModelArts agency.

1. Log in to the ModelArts management console. In the navigation pane on the left, choose **Permission Management**. On the displayed page, locate the target agency, choose **View Permissions** in the **Operation** column, and click **Modify permission in IAM**.
2. On the IAM console, choose **Agencies** from the navigation pane on the left, and choose **Permissions** > **Authorize**. Search for **IAM ReadOnlyAccess**, enable it, and click **Next** and **OK**.

Step 6 Verify that the permission is granted.

Log in to ModelArts as the IAM user, only the configured SFS Turbo folders are displayed during training job creation and notebook creation.

----End

10.7 FAQ

10.7.1 What Do I Do If a Message Indicating Insufficient Permissions Is Displayed When I Use ModelArts?

If a message indicating insufficient permissions is displayed when you use ModelArts, perform the operations described in this section to grant permissions for related services as needed and check the user permissions.

This section uses insufficient OBS permissions as an example to describe how to grant OBS permissions to a user. It also offers reference for other permission insufficiency scenarios. The only difference lies in the authorization scope. For details about the authorization scope in different scenarios, see **Agencies and Dependencies**.

The permissions to use ModelArts depend on OBS authorization. Therefore, ModelArts users require OBS system permissions as well.

- For details about how to grant a user full permissions for OBS and common operations permissions for ModelArts, see **Configuring Common Operations Permissions**.
- For details about how to manage user permissions on OBS and ModelArts in a refined manner and configure custom policies, see **Creating a Custom Policy for ModelArts**.

Configuring Common Operations Permissions

To use ModelArts basic functions, assign the **ModelArts CommonOperations** permission on project-level services to users. Since ModelArts depends on OBS permissions, you also need to log in to the IAM console and assign the **OBS Administrator** permission on global services to users.

The procedure is as follows:

Step 1 Create a user group.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Log in to the IAM console. Choose **User Groups** and click **Create User Group**. Enter a user group name, and click **OK**.

Step 2 Configure permissions for the user group.

In the user group list, locate the user group created in step 1, click **Authorize**, and perform the following operations.

1. Assign the **ModelArts CommonOperations** permission on project-level services to the user group and click **OK**.

   NOTE

   The permission takes effect only in assigned regions. Assign permissions in all regions if the permission is required in all regions.

2. Assign the **OBS Administrator** permission on global services to the user group and click **OK**.

Step 3 **Create a user and add it to a user group.**

Create a user on the IAM console and add the user to the user group created in step 1.

Step 4 **Log in as the IAM user** and verify permissions.

Log in to the ModelArts console as the created user, switch to the authorized region, and verify the **ModelArts CommonOperations** and **Tenant Administrator** policies are in effect.

- Choose **Service List** > **ModelArts**. In the navigation pane of the ModelArts console, choose your desired type of AI dedicated resource pools and create one. You should not be able to create a new resource pool if the **ModelArts CommonOperations** permission has taken effect.
- Choose any other service in **Service List**. (Assume that the current policy contains only **ModelArts CommonOperations**.) If a message appears indicating that you have insufficient permissions to access the service, the **ModelArts CommonOperations** policy has already taken effect.
- Choose **Service List** > **ModelArts**. On the ModelArts console, choose **Data Management** > **Datasets**. Click **Create Dataset**. You should be able to access the corresponding OBS path.

----End

Creating a Custom Policy for ModelArts

In addition to the default system policies of ModelArts, you can create custom policies, which can address OBS permissions as well. For more information, see **Creating a Custom Policy**.

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

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

NOTE

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

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

1. Create a custom policy for minimizing permissions for OBS that ModelArts depends on.

   Log in to the IAM console, choose **Permissions** > **Policies/Roles**, and click **Create Custom Policy**. Configure the parameters as follows:

   - **Policy Name**: Choose a custom policy name.
   - **Policy View**: **JSON**
   - **Policy Content**: Follow the instructions in **Example Custom Policies of OBS**. For more information about OBS system permissions, see **OBS Permissions Management**.

2. Create a custom policy for the permissions to use ModelArts development environments. Configure the parameters as follows:

   - **Policy Name**: Choose a custom policy name.
   - **Policy View**: **JSON**
   - **Policy Content**: Follow the instructions in **Example Custom Policies for Using the ModelArts Development Environment**. For the actions that can be added for custom policies, see **ModelArts API Reference > Permissions Policies and Supported Actions**.
   - For the system policies of other services, see **System Permissions**.

3. **Create a user group and grant permissions to the user group** on the IAM console.

   After creating a user group on the IAM console, grant the custom policy created in **1** to the user group.

4. **Create a user and add it to a user group.**

   Create a user on the IAM console and add the user to the group created in **3**.

5. **Log in as the IAM user** and verify permissions.

   Log in to the ModelArts console as the created user, switch to the authorized region, and verify the **ModelArts CommonOperations** and **Tenant Administrator** policies are in effect.

   Choose **Service List** > **ModelArts**.

   - In the navigation pane, choose **Asset Management** > **Datasets**. If you cannot create a dataset (you currently only have permissions for the development environment), it indicates that the policy granting the user permissions limited to the ModelArts development environment has taken effect.
   - In the navigation pane on the left, perform the following operations:
     - New version: Choose **Model Build** > **Notebook**.
     - Old version: Choose **Development Workspace** > **Notebook**.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

In the upper right corner of the notebook page, click **Create**. If you can access the OBS path specified in **Storage Settings**, the configured OBS permissions have taken effect.

Example Custom Policies of OBS

The permissions to use ModelArts require OBS authorization. The following example shows the minimum OBS required, including the permissions for OBS buckets and objects. After being granted the minimum permissions for OBS, users can access OBS from ModelArts without restrictions.

```
{
   "Version": "1.1",
   "Statement": [
      {
         "Action": [
            "obs:bucket:ListAllMybuckets",
            "obs:bucket:HeadBucket",
            "obs:bucket:ListBucket",
            "obs:bucket:GetBucketLocation",
            "obs:object:GetObject",
            "obs:object:GetObjectVersion",
            "obs:object:PutObject",
            "obs:object:DeleteObject",
            "obs:object:DeleteObjectVersion",
            "obs:object:ListMultipartUploadParts",
            "obs:object:AbortMultipartUpload",
            "obs:object:GetObjectAcl",
            "obs:object:GetObjectVersionAcl",
            "obs:bucket:PutBucketAcl",
            "obs:object:PutObjectAcl"
         ],
         "Effect": "Allow"
      }
   ]
}
```

Example Custom Policies for Using the ModelArts Development Environment

```
{
   "Version": "1.1",
   "Statement": [

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

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

11 Best Practices of Security Configuration

Scenario

Security is a shared responsibility between Huawei Cloud and yourself. Huawei Cloud ensures the security of cloud services for a secure cloud. As a tenant, you should properly use the security capabilities provided by cloud services to protect data and securely use the cloud.

This section provides actionable guidance for enhancing the overall security of ModelArts. You can continuously evaluate the security status of your ModelArts resources and enhance their overall security by combining different security capabilities provided by ModelArts. By doing this, data stored in ModelArts can be protected from leakage and tampering both at rest and in transit.

Consider the following aspects for your security configurations:

- **Using a Dedicated Resource Pool in the Production Environment**
- **Running a Custom Image as a Non-root User**
- **Not Using Hard-coded Credentials During Development**
- **Using Independent Agencies for Different IAM Users**

Using a Dedicated Resource Pool in the Production Environment

When you use the training, inference, and development environments, you shall use the dedicated resource pool in the production environment, which provides exclusive compute resources and enhanced secure resource isolation capabilities. For details about how to use a dedicated resource pool, see **Creating a Dedicated Resource Pool**.

When using ModelArts for full-process AI development, you can use two different resource pools.

Public resource pools: provide large-scale public compute clusters, which are allocated based on job parameter settings. Resources are isolated by job. You will be billed based on resource specifications, usage duration, and the number of instances used in a public resource pool, regardless of tasks (training, deployment, or development). Public resource pools are provided by ModelArts by default and

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

do not need to be created or configured. You can directly select a public resource pool during AI development.

Dedicated resource pools: provide dedicated compute resources, which can be used for notebook instances, training jobs, and model deployment. The resources provided in a dedicated resource pool are exclusive, featuring higher resource efficiency than a public resource pool.

To use a dedicated resource pool, you need to purchase one and select it during AI development.

Running a Custom Image as a Non-root User

You can create a Dockerfile for a custom image and then push it to SWR. To enhance permission control, you shall explicitly define the default running user as a non-root user when customizing an image. This helps reduce security risks during container runtime.

During the development and runtime of AI services, complex environment dependencies need to be debugged for solidifying configurations. In the best practices of AI development in ModelArts, container images are used to solidify the runtime environment. In this way, dependencies can be managed and the runtime environment can be easily switched. ModelArts offers cloud-based container resources that speed up AI development and streamline model testing cycles.

For details about how to use custom images in ModelArts, see **Image Functions in ModelArts**.

Not Using Hard-coded Credentials During Development

If you want to develop an algorithm and publish it to the production environment in ModelArts Notebook, you shall check the password, AK/SK, database connection, OBS connection, and SWR connection information used in the code. Do not use fixed authentication credentials to facilitate subsequent algorithm update and maintenance. You shall encrypt the preceding sensitive information and save it in the program configuration file.

Using Independent Agencies for Different IAM Users

To use ModelArts resources, ensure that you have obtained the agency authorization from the user. For better permission control over IAM users, you shall grant agency permissions to each IAM user individually on the ModelArts global configuration page. The same agency credential shall not be shared by multiple IAM users. For details about agency authorization, see **Creating an IAM User and Granting ModelArts Permissions**.

Using Content Security Services in Generative AI

When deploying generative AI in the production environment, you need to pay attention to the varying regulatory requirements for generative AI models and applications across different regions. For example, AI-generated content might contravene fundamental societal values or touch upon locally sensitive political issues.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

You must safeguard AI-generated content by deploying content security services tailored to your needs, which can filter and manage content to minimize outputs from generative AI models that conflict with local values, are harmful, or breach regional laws. Additionally, it is essential to notify local authorities prior to launching operational systems to secure necessary permissions.

To address the handling of AI-generated content, refer to content security services provided by third-party vendors. Enabling APIs allows for processing the generated content effectively. These services offer foundational content recognition and management functionalities, enabling customization according to specific production system requirements.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.