ModelArts
7.2.0
FAQs

Issue	01
Date	2025-10-24

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 Permissions

1.1 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.

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.

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**. Choose **Dedicated Resource Pools**. On the page that is displayed, select a resource pool type and click **Create**. You should not be able to create a new resource pool.
- 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** > **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.

NOTE

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

ModelArts is a project-level service, but OBS is a global service, 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**. On the ModelArts console, choose **Data Management** > **Datasets**. If you cannot create a dataset, the permissions (for using the development environment) granted only to ModelArts users have taken effect.
   - Choose **Service List** > **ModelArts**. On the ModelArts console, choose **DevEnviron** > **Notebook** and click **Create**. If you can access the OBS path specified in **Storage**, the 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"
      ]
    }
  ]
}
```

1.2 How Do I Isolate IAM Users on a Notebook Instance?

In a development environment, multiple IAM users may require isolation, and they don't want their notebook instances to be viewed, modified, or deleted by others.

Currently, there are two solutions to isolating IAM users:

- Solution 1: Delete the **modelarts:notebook:listAllNotebooks** permission.
- Solution 2: Use **workspaces**. As an enterprise user, you can submit the request for enabling the workspace function to your technical support.

1.3 How Do I Obtain an Access Key?

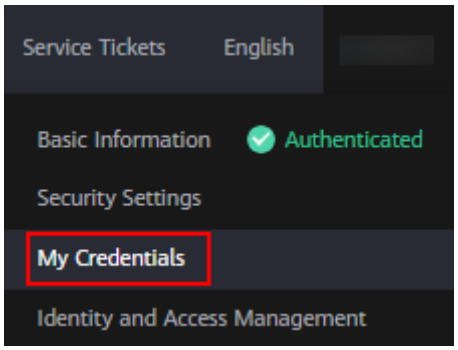
Obtaining an Access Key

1. Log in to **HUAWEI CLOUD** and click **Console** in the upper right corner of the page to access the HUAWEI CLOUD management console.

Figure 1-1 Console

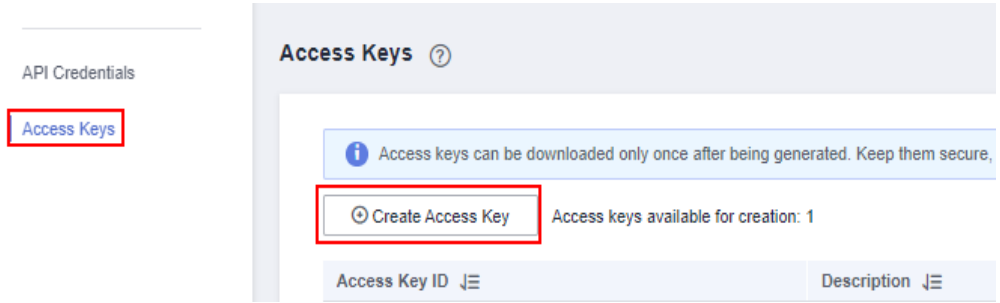
2. Hover the cursor over the account name in the upper right corner of the console and choose **My Credentials** from the drop-down list. The **API Credentials** page is displayed.

Figure 1-2 My Credentials



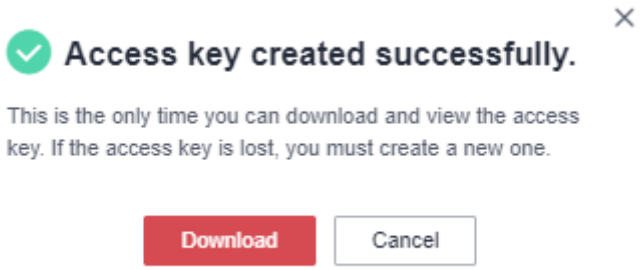
3. On the **API Credentials** page, choose **Access Keys** > **Create Access Key**.

Figure 1-3 Create Access Key



4. Enter the description of the key and click **OK**. Click **Download** to download the key.

Figure 1-4 Access key created



5. The access key file is saved in the default downloads folder of the browser. Open the **credentials.csv** file to view the access key with an AK and SK.

2 Storage

2.1 How Do I View All Files Stored in OBS on ModelArts?

To view all files stored in OBS when using notebook instances or training jobs, use either of the following methods:

- OBS console

  Log in to OBS console using the current account, and search for the OBS buckets, folders, and files.

- You can use an API to check whether a given directory exists. In an existing notebook instance or after creating a new notebook instance, run the following command to check whether the directory exists:

  ```
  import moxing as mox
  mox.file.list_directory('obs://bucket_name', recursive=True)
  ```

  If there are a large number of files, wait until the final file path is displayed.

3 Standard Workflow

3.1 How Do I Locate Workflow Running Errors?

If a workflow reports an error when it runs in run mode, follow these steps:

1. Check that you have installed the latest version of the SDK package to avoid package version inconsistency.
2. Check that your SDK code follows the specifications. See the corresponding code sample for details.
3. Check that the content input during the execution is correct and matches the format required in the prompt message.
4. Find the code line where the error occurs based on the error information and analyze the context logic.

Common Errors in Historical SDK Packages

- Error reported the service deployment phase

```
'weight': 100, 'specification':'custom',
'cluster_id':'******','custom_spec':{'cpu':2.5, 'memory':
1024},'envs':{'******': '******', '******': '******'}}]"

Note that:(1) The "[" at the beginning and "]" at the end
are required.
          (2) The sum of the weights must be equal to 100.
          (3) All model must have the same model name. Two
model versions cannot be the same.
[{'model_name':'      :','model_version':'      ','specific
ation':'                    ','weight':100}]
Traceback (most recent call last):
```

- Error reported after service parameters are entered

```
    return self.name == obj.name and \

AttributeError: 'str' object has no attribute 'name'
```


Solution

Rectify these errors by upgrading the SDK to the latest version.

4 ModelArts Standard Data Preparation

4.1 Is There a File Size Limit for Images to Be Added to a ModelArts Dataset?

Data management sets a size limit for images to be uploaded to datasets used for object detection or image classification labeling tasks. The size of an image cannot exceed 8 MB, and only JPG, JPEG, PNG, and BMP formats are supported.

Solutions:

- Import the images from OBS. Upload images to any OBS directory and import the images from the OBS directory to an existing dataset.
- Use data source synchronization. Upload images to the input directory or its subdirectory of a dataset, and click **Synchronize Data Source** on the dataset details page to add new images. Note that synchronizing a data source will delete the files deleted from OBS from the dataset. Exercise caution when performing this operation.
- Create a dataset. Upload images to any OBS directory. You can directly use the image directory as the input directory to create the dataset.

4.2 How Do I Import Local Labeled Data to ModelArts?

ModelArts allows you to import data by importing datasets. Locally labeled data can be imported from an OBS directory or the manifest file. After the import, you can label the data again or modify the labels in ModelArts Data Management.

For details about how to import data from an OBS directory or a manifest file, see **Importing Data**.

4.3 Where Are the Data Labeling Results Stored in ModelArts?

The ModelArts console provides data visualization, which allows you to view detailed data and labeling information on the console. To learn more about the path for storing labeling results, see the following description.

Background

When creating a dataset in ModelArts, set both **Input Dataset Path** and **Output Dataset Path** to OBS.

- **Input Dataset Path**: OBS path where the raw data is stored.
- **Output Dataset Path**: Under this path, directories are generated based on the dataset version after data is labeled in ModelArts and datasets are published. The manifest files (containing data and labeling information) used in ModelArts are also stored in this path. For details about the files, see **Directory Structure of Dataset Versions**.

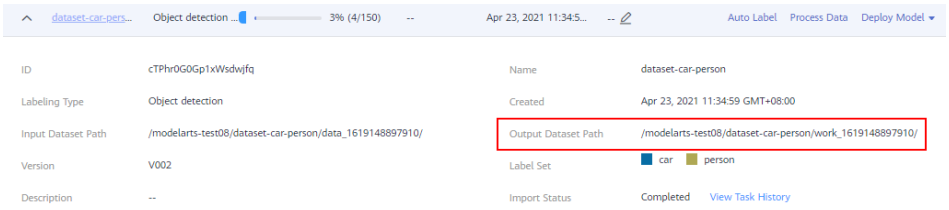
Procedure

1. Log in to the ModelArts console and choose **Data Management** > **Datasets**.
2. Select your desired dataset and click the triangle icon on the left of the dataset name to expand the dataset details. You can obtain the OBS path set for **Output Dataset Path**.

   NOTE

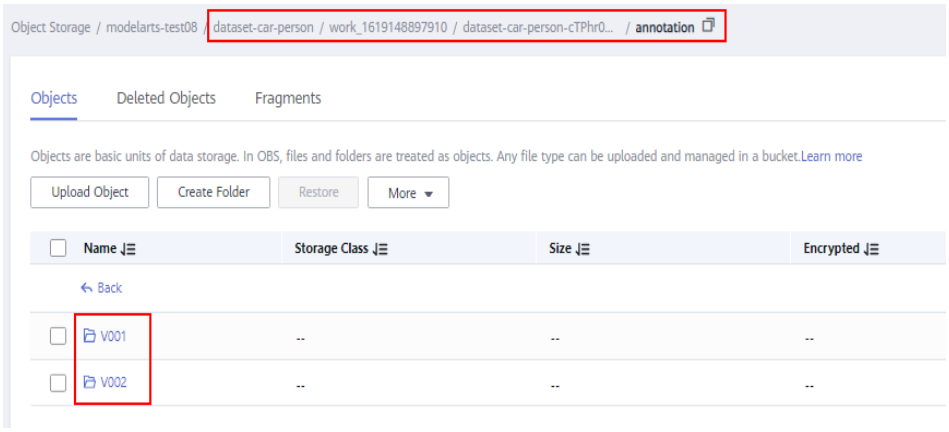
   Before obtaining labeling results, ensure that at least one dataset version is available.

   Figure 4-1 Dataset details

3. Log in to the OBS console and locate the directory of the corresponding dataset version from the OBS path obtained in **2** to obtain the labeling result of the dataset.

Figure 4-2 Obtaining the labeling result



4.4 How Do I Download Labeling Results from ModelArts to a Local PC?

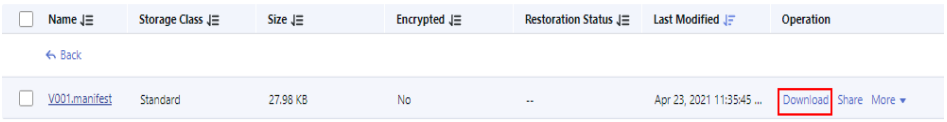
After being published, the labeling information and data in ModelArts datasets are stored as manifest files in the OBS path set for **Output Dataset Path**.

To obtain the OBS path, do as follows:

1. Log in to the ModelArts management console and choose **Data Management** > **Datasets**.
2. Locate the target dataset and click the triangle icon on the left of the dataset name to expand the dataset details. You can obtain the OBS path set for **Output Dataset Path**.
3. Log in to the OBS management console and locate the version directory from the obtained OBS path to obtain the labeling result of the dataset.

To download the labeling results to a local PC, go to the OBS path where the manifest files are stored and click **Download**.

Figure 4-3 Downloading labeling results



4.5 Why Can't Team Members Receive Emails for a Team Labeling Task in ModelArts?

The possible causes are as follows:

- All dataset data has been labeled. An email can be sent to team members only if there is unlabeled data in the dataset when the team labeling task is created.
- Team members receive emails for team labeling tasks. No email will be sent when you create a labeling team or add members to a labeling team.

- Your email address has not been configured or has been incorrectly configured. For details about how to configure an email address, see **Managing Team Members**.
- Team members' email addresses are blocked.

4.6 How Is Data Distributed Between Team Members During Team Labeling in ModelArts?

Data is evenly assigned to all members in a labeling team, but not assigned to certain members.

- If the data cannot be evenly distributed because the number of images is not in proportion to the number of team members, the excessive images will be randomly distributed to team members.
- If the number of samples is less than the number of team members, some members cannot receive any samples. Samples are allocated only to annotators. For example, there are 10,000 samples to be labeled and all the five members in the team are annotators, each annotator will be allocated with 2000 samples.

4.7 How Do I Merge Two Datasets in ModelArts?

Datasets cannot be merged directly.

However, you can perform the following operations to merge the data of two datasets into a new dataset.

For example, to merge datasets A and B, do the following:

1. Publish datasets A and B.
2. Obtain the manifest files of the two datasets from the OBS path set for **Output Dataset Path**.
3. Create an empty dataset C and select an empty OBS folder for **Input Dataset Path**.
4. Import the manifest files of datasets A and B to dataset C.

   After the import is complete, data in datasets A and B is merged into dataset C. Publish dataset C so it can be used.

4.8 Why Are Images Displayed in Different Angles Under the Same Account in ModelArts?

There are rotation angles of certain images, and the rules of processing such images vary depending on browsers. The following figures show compatibility with browsers.

- L indicates the latest version. L3 indicates the latest three stable browser versions when the product is released.
- If your browser is of an earlier version, the page display will be adversely affected, and the system will prompt you to upgrade your browser.

- If your browser is not compatible with the management console, the system will advise you to upgrade your browser or install a desired browser.

Table 4-1 Compatibility with PC browsers

Browser	Version	OS	Compatibility
Internet Explorer	11	Windows 7	Not guaranteed
Microsoft Edge	L3	Windows 10	Fully compatible
	< 79	Windows 10	Not guaranteed
Mozilla Firefox	L3	Windows 10	Fully compatible
	L3	CentOS 7+	Partially compatible You can use this version to perform basic interactive operations, but visual and interactive effects may be affected.
	L3	Ubuntu 14.04 LTS+	Partially compatible You can use this version to perform basic interactive operations, but visual and interactive effects may be affected.
	L3	macOS 10+	Partially compatible You can use this version to perform basic interactive operations, but visual and interactive effects may be affected.
Google Chrome	L3	Windows 10	Fully compatible
	L3	CentOS 7+	Partially compatible You can use this version to perform basic interactive operations, but visual and interactive effects may be affected.
	L3	Ubuntu 14.04 LTS+	Partially compatible You can use this version to perform basic interactive operations, but visual and interactive effects may be affected.

Browser	Version	OS	Compatibility
	L3	macOS 10+	Partially compatible You can use this version to perform basic interactive operations, but visual and interactive effects may be affected.
Safari	L2	macOS 10+	Partially compatible You can use this version to perform basic interactive operations, but visual and interactive effects may be affected.

Table 4-2 Compatibility with mobile browsers

Browser	Version	OS	Compatibility
Chrome	L3	Android	Fully compatible
Safari	L3	IOS	Fully compatible
UC Browser	L3	Android	Fully compatible
QQ Browser	L3	Android	Fully compatible
360 Secure Browser	L3	Android	Fully compatible
Baidu Browser	L3	Android	Fully compatible

4.9 Do I Need to Train the Model Again Using the Data Newly Added After Auto Labeling Is Complete in ModelArts?

After auto labeling is complete, you need to confirm the labeling result.

- If new data is added with the labeling result unconfirmed, auto labeling will be performed again, and you need to retrain the model using both the unconfirmed data and the new data.
- If new data is added after the labeling result is confirmed, you need to retrain the model using only the new data.

4.10 How Do I Split an Image Dataset into Training and Validation Sets in ModelArts?

ModelArts does not support manual addition of images to a training or validation dataset, but allows you to set split ratios for training and validation sets. The system randomly allocates the images to the training and validation datasets based on the configured ratios.

Setting Split Ratios

During dataset publishing, you can configure data splitting only for datasets used for image classification, object detection, text classification, or sound classification tasks.

By default, data splitting is disabled. After this function is enabled, set split ratios.

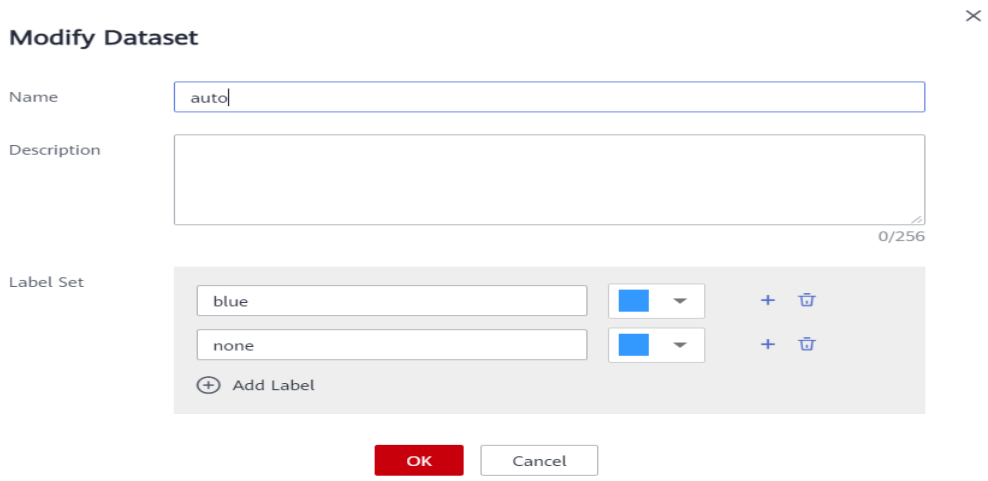
Enter a value ranging from 0 to 1 for the training set ratio. After the training set ratio is set, the validation set ratio is automatically filled. The sum of the training set ratio and the validation set ratio is 1.

The training set ratio is the ratio of sample data used for model training. The validation set ratio is the ratio of the sample data used for model validation. The training and validation ratios affect the performance of training templates.

4.11 Can I Customize Labels During Object Detection Labeling in ModelArts?

Yes. You can add custom labels to the label set of an object detection dataset by modifying the dataset.

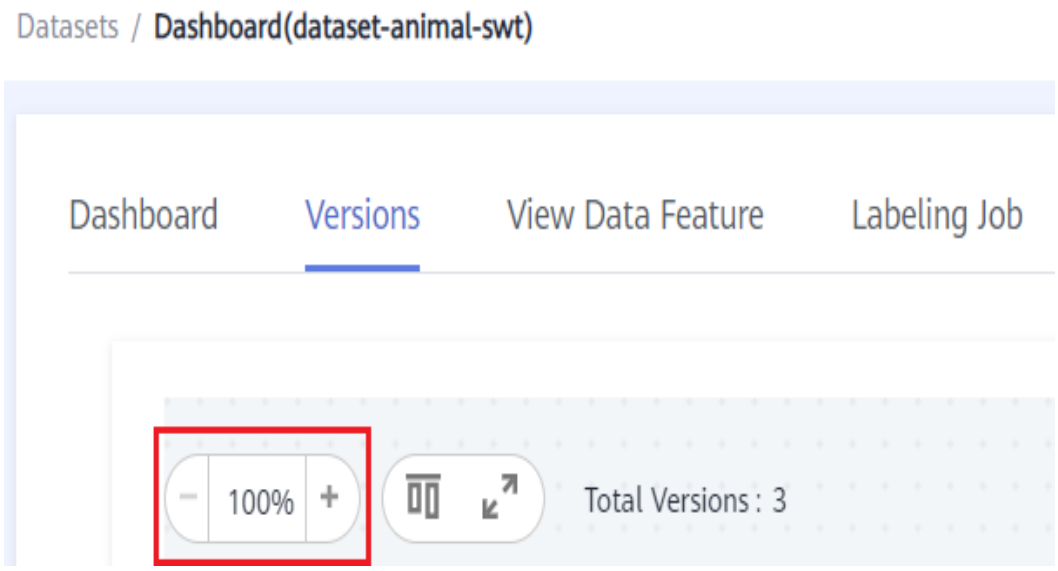
Figure 4-4 Modify Dataset



4.12 What Should I Do If I Can't Find a New Dataset Version in ModelArts?

The version list page can be zoomed in or out. Zoom out the page before searching.

Click the name of the target dataset to go to the dataset overview page. Then, zoom out the **Versions** page.



4.13 How Do I Split a Dataset in ModelArts?

During dataset publishing, you can configure data splitting only for datasets used for image classification, object detection, text classification, or sound classification tasks.

By default, data splitting is disabled. After this function is enabled, set split ratios.

Enter a value ranging from 0 to 1 for the training set ratio. After the training set ratio is set, the validation set ratio is automatically filled. The sum of the training set ratio and the validation set ratio is 1.

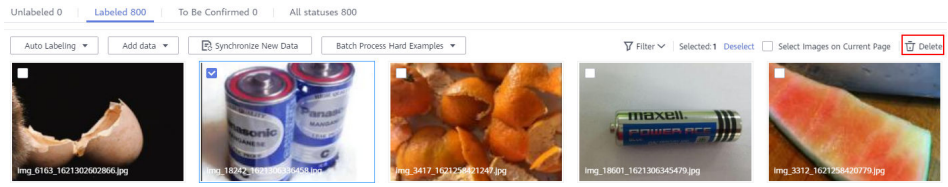
The training set ratio is the ratio of sample data used for model training. The validation set ratio is the ratio of the sample data used for model validation. The training and validation ratios affect the performance of training templates.

4.14 How Do I Delete Images from a Dataset in ModelArts?

1. Log in to the ModelArts management console. In the navigation pane, choose **Data Management** > **Label Data**. The data labeling task list is displayed. Click the dataset from which you want to delete images. The labeling details page is displayed.

2. On the **All statuses**, **Unlabeled**, or **Labeled** tab page, select the images to be deleted or click **Select Images on Current Page**, and click to delete them. In the displayed dialog box, select or deselect **Delete the source files from OBS** as required. After confirmation, click **Yes** to delete the images.
   If a tick is displayed in the upper left corner of an image, the image is selected. If no image is selected on the page, is unavailable.

   Figure 4-5 Deleting an image from a dataset



4.15 What Do I Do If Images in a Dataset Cannot Be Displayed?

Symptom

Images in a created dataset cannot be displayed during labeling, and they cannot be viewed by clicking them. Alternatively, the system displays a message indicating that an error occurred in image loading.

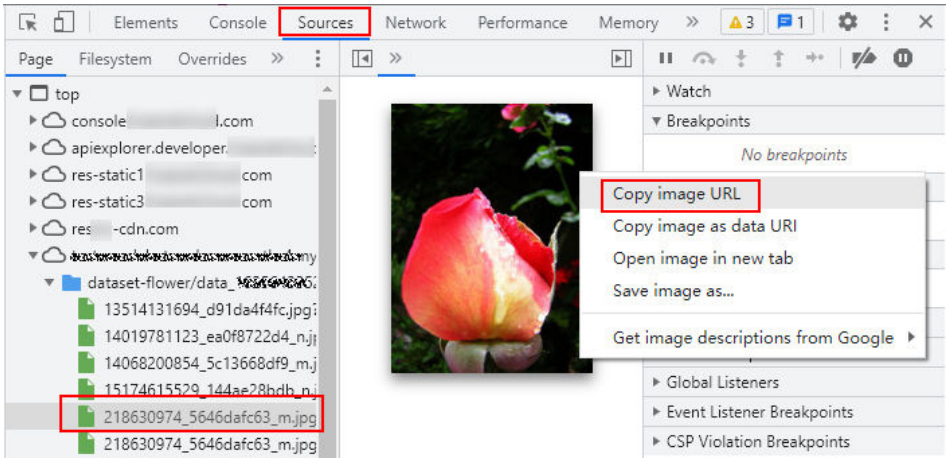
Possible Cause

- The local network may be faulty. As a result, OBS cannot be accessed and images cannot be loaded.
- You are not allowed to access the target OBS bucket.
- The OBS bucket or file may be encrypted.
- The OBS storage class does not allow the parallel file system to process images. Therefore, the thumbnails cannot be displayed.

Solution

1. The following uses Google Chrome as an example. Press **F12** to open the browser console, locate the image, and copy the image URL.

Figure 4-6 Obtaining the image URL



2. Enter the URL in a new browser. The "Your Connection Is Not Private" message is displayed. Click **Advanced** on the page and choose **Proceed to <link> (unsafe)** to go to the target website.
3. After the image is successfully accessed, return to the ModelArts console to access the dataset. The image is displayed.

5 ModelArts Standard Notebook

5.1 Is the Keras Engine Supported by ModelArts Notebook Instances?

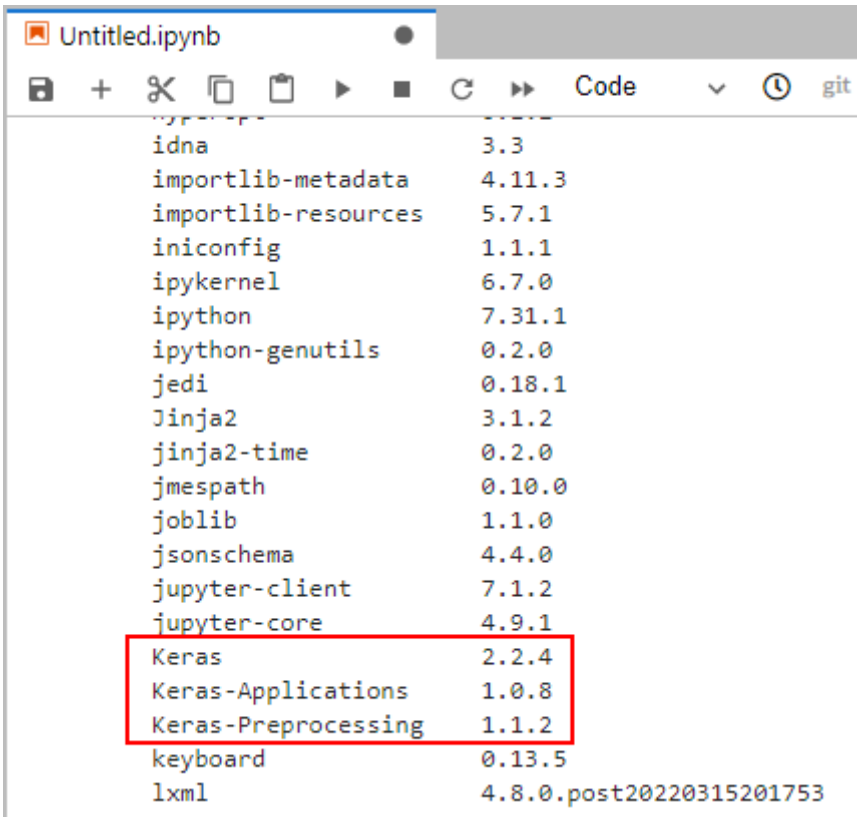
Notebook instances in **DevEnviron** support the Keras engine. The Keras engine is not supported in job training and model deployment (inference).

Keras is an advanced neural network API written in Python. It is capable of running on top of TensorFlow, CNTK, or Theano. Notebook instances in **DevEnviron** support **tf.keras**.

Viewing Keras Versions

1. On the ModelArts management console, create a notebook instance with image **TensorFlow-1.13** or **TensorFlow-1.15**.
2. Access the notebook instance. In JupyterLab, run **!pip list** to view Keras versions.

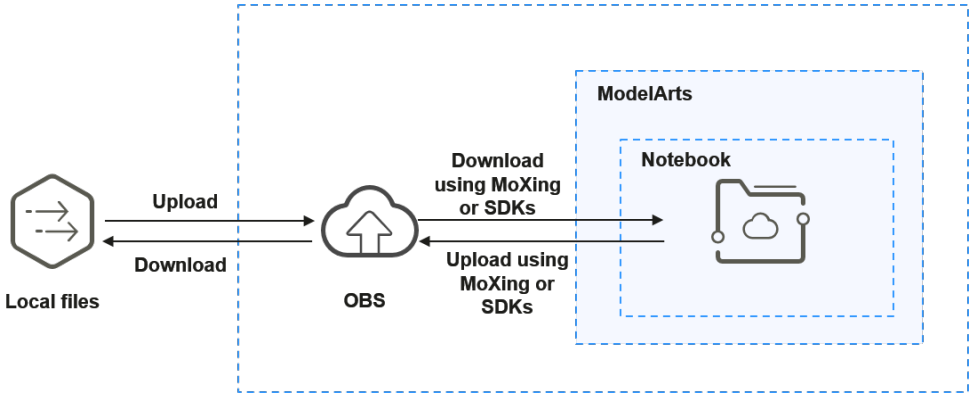
Figure 5-1 Viewing Keras versions



5.2 How Do I Upload a File from a Notebook Instance to OBS or Download a File from OBS to a Notebook Instance in ModelArts?

In a notebook instance, you can call the ModelArts MoXing API or SDK to exchange data with OBS for uploading a file to OBS or downloading a file from OBS to the notebook instance.

Figure 5-2 Uploading or downloading a file



For details about how to upload files using OBS Browser, see **Uploading and Downloading Files Through OBS Browser+**.

Method 1: Using MoXing to Upload and Download a File

Developed by the ModelArts team, MoXing is a distributed training acceleration framework built on open-source deep learning engines such as TensorFlow and PyTorch. MoXing makes model coding easier and more efficient.

MoXing provides a set of file object APIs for reading and writing OBS files.

For details about the mapping between MoXing APIs and native APIs and how to call APIs, see **MoXing File Operations**.

Sample code:

```
import moxing as mox

# Download the OBS folder sub_dir_0 from OBS to a notebook instance.
mox.file.copy_parallel('obs://bucket_name/sub_dir_0', '/home/ma-user/work/sub_dir_0')
# Download the OBS file obs_file.txt from OBS to a notebook instance.
mox.file.copy('obs://bucket_name/obs_file.txt', '/home/ma-user/work/obs_file.txt')

# Upload the OBS folder sub_dir_0 from a notebook instance to OBS.
mox.file.copy_parallel('/home/ma-user/work/sub_dir_0', 'obs://bucket_name/sub_dir_0')
# Upload the OBS file obs_file.txt from a notebook instance to OBS.
mox.file.copy('/home/ma-user/work/obs_file.txt', 'obs://bucket_name/obs_file.txt')
```

Method 2: Using SDK to Upload and Download a File

Call the ModelArts SDK for downloading a file from OBS.

Sample code: Download **file1.txt** from OBS to **/home/ma-user/work/** in the notebook instance. All the bucket name, folder name, and file name are customizable.

```
from modelarts.session import Session
session = Session()
session.obs.download_file(src_obs_file="obs://bucket-name/dir1/file1.txt", dst_local_dir="/home/ma-user/
work/")
```

Call the ModelArts SDK for downloading a folder from OBS.

Sample code: Download **dir1** from OBS to **/home/ma-user/work/** in the notebook instance. The bucket name and folder name are customizable.

```
from modelarts.session import Session
session = Session()
session.obs.download_dir(src_obs_dir="obs://bucket-name/dir1/", dst_local_dir="/home/ma-user/work/")
```

Call the ModelArts SDK for uploading a file to OBS.

Sample code: Upload **file1.txt** in the notebook instance to OBS bucket **obs://bucket-name/dir1/**. All the bucket name, folder name, and file name are customizable.

```
from modelarts.session import Session
session = Session()
session.obs.upload_file(src_local_file='/home/ma-user/work/file1.txt', dst_obs_dir='obs://bucket-name/dir1/')
```

Call the ModelArts SDK for uploading a folder to OBS.

Sample code: Upload **/work/** in the notebook instance to **obs://bucket-name/dir1/work/** of **bucket-name**. The bucket name and folder name are customizable.

```
from modelarts.session import Session
session = Session()
session.obs.upload_dir(src_local_dir='/home/ma-user/work/', dst_obs_dir='obs://bucket-name/dir1/')
```


Error Handling

If you download a file from OBS to your notebook instance and the system displays error message "Permission denied", perform the following operations for troubleshooting:

- Ensure that the target OBS bucket and notebook instance are in the same region, for example, **CN North-Beijing4**. If the OBS bucket and notebook instance are in different regions, the access to OBS is denied. For details, see **Check whether the OBS bucket and ModelArts are in the same region**.
- Ensure that your account for performing notebook operations has the permission to read data in the OBS bucket. If you do not have the permission, see **How Do I Access the OBS Bucket of Another Account from a ModelArts Notebook Instance?**.

5.3 Where Is Data Uploaded from a ModelArts Notebook Instance?

Data may be stored in OBS or EVS, depending on which kind of storage you have configured for your Notebook instances:

- OBS

  After you click **upload**, the data is directly uploaded to the target OBS path specified when the notebook instance was created.
- EVS

  After you click **upload**, the data is uploaded to the instance container, that is, the **~/work** directory on the **Terminal** page.

5.4 How Do I Copy Data from Notebook A to Notebook B in ModelArts?

Data cannot be directly copied from notebook A to notebook B. To copy data, do as follows:

1. Upload the data of notebook A to OBS.
2. Download data from OBS to notebook B.

For details about how to upload and download files, see **How Do I Upload a File from a Notebook Instance to OBS or Download a File from OBS to a Notebook Instance in ModelArts?**

5.5 How Do I Rename an OBS File on a ModelArts Notebook Instance?

OBS files cannot be renamed on the OBS console. To rename an OBS file, call a MoXing API in an existing or newly created notebook instance.

The following is an example:

Rename **obs_file.txt** to **obs_file_2.txt**.

```
import moxing as mox
mox.file.rename('obs://bucket_name/obs_file.txt', 'obs://bucket_name/obs_file_2.txt')
```

5.6 How Do I Use the pandas Library to Process Data in OBS Buckets on a ModelArts Notebook Instance?

Step 1 Download data from OBS to a notebook instance. For details, see **Downloading a File from JupyterLab to a Local Path**.

Step 2 Process pandas data by following the instructions provided in ***pandas User Guide***.

----End

5.7 How Do I Access the OBS Bucket of Another Account from a ModelArts Notebook Instance?

If you select OBS when creating a notebook instance, you can access only the OBS buckets of your own account.

To access OBS files of another account from a notebook instance, you must have read and write permissions for the target OBS bucket.

1. Contact the creator of the OBS bucket to grant the read and write permissions for the OBS bucket to the current account. For details, see **Granting Other Accounts the Read/Write Permission for a Bucket**. This section describes how a Huawei Cloud account grants its OBS bucket permissions to other Huawei Cloud accounts. If you are an IAM user or in other scenarios, see section "Configuration Cases in Typical Permission Control Scenarios" in **Object Storage Service Permissions Configuration Guide** for instructions about how to grant OBS bucket permissions.
2. After obtaining read and write permissions for the OBS bucket, use the MoXing API on your notebook instance to access the OBS bucket and read its data. Example command:

   ```
   import moxing as mox
   mox.file.copy_parallel('obs://bucket_1/dataset', 'obs://bucket_2/dataset')
   ```

   bucket_1 indicates the OBS bucket of another account, and **bucket_2** indicates your own OBS bucket.

5.8 What Is the Default Working Directory of JupyterLab on ModelArts Notebook Instances?

- **OBS**

  For this type of notebook instances, the files uploaded to JupyterLab are stored in the OBS path specified during the instance creation by default.

  All read and write operations are performed on the files selected in the OBS path and are irrelevant to the current instance space. To synchronize data from an OBS path to the instance space, use **JupyterLab upload and download functions**.

- **Notebook instance with EVS storage**

  For this type of notebook instances, the files uploaded to JupyterLab are by default stored in the EVS space automatically allocated during the instance creation.

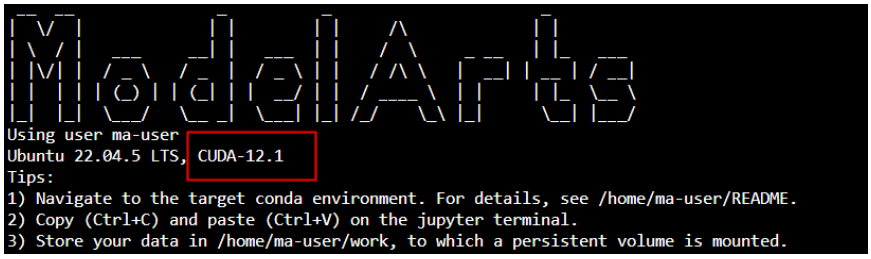
  All read and write operations are performed on the files selected in the EVS space. You can mount your data to the **~/work** directory.

5.9 How Do I Check the CUDA Version Used by a ModelArts Notebook Instance?

1. Log in to the ModelArts console and choose **Development Workspace** > **Notebooks** from the navigation pane.
2. In the notebook list, click **Open** in the **Operation** column of the target notebook.
3. On the **Launcher** page, click **Terminal**. The CUDA version is displayed by default.

   As shown in the following figure, the CUDA version is 12.1.

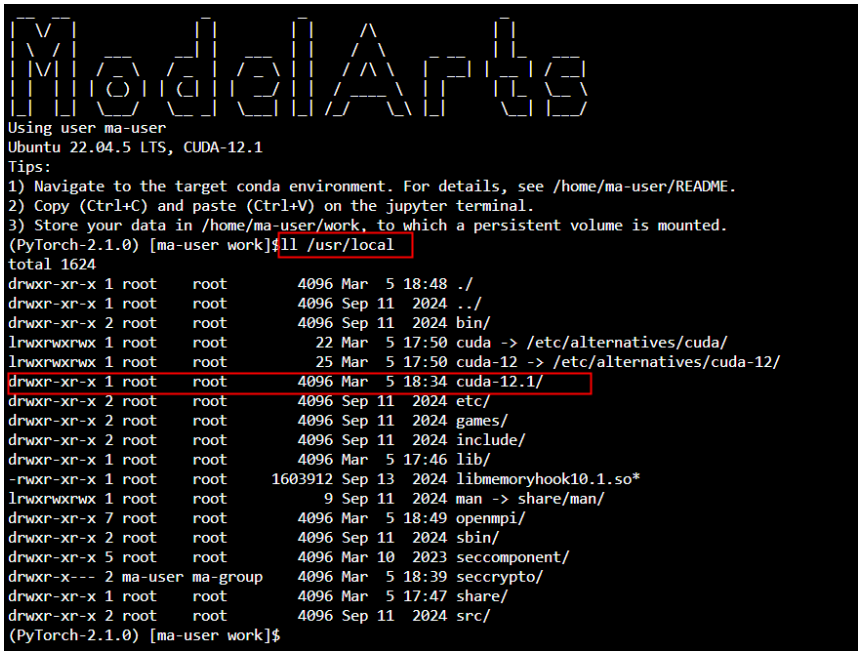
   Figure 5-3 Checking the CUDA version



   You can also run the following command to check the CUDA version in the environment:

```
ll /usr/local
```

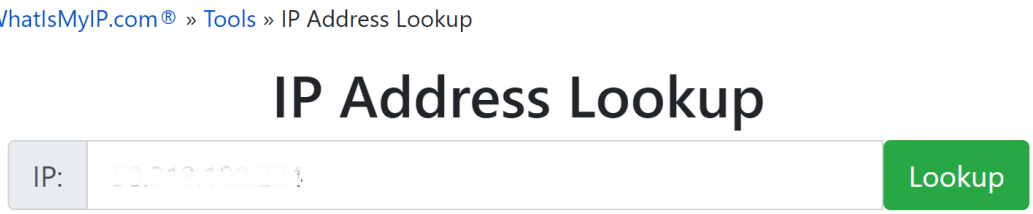

Figure 5-4 Running the command to check the CUDA version



5.10 How Do I Obtain the External IP Address of the Local Host from a ModelArts Notebook Instance?

Search for "IP address lookup" in a mainstream search engine.

Figure 5-5 IP address lookup



5.11 Is There a Proxy for ModelArts Notebook Instances? How Do I Disable It?

There is a proxy for Notebook.

Run the **env|grep proxy** command to obtain the notebook proxy.

Run the **unset https_proxy unset http_proxy** command to disable the proxy.

5.12 How Do I Customize Engine IPython Kernel If the Built-in Engines of ModelArts Notebook Instances Do Not Meet My Requirements?

Scenarios

If the default engine environment in notebook does not meet your needs, you can create a conda environment to set up your own environment as required. This section uses python3.6.5 and tensorflow1.2.0 as an example to describe how to set up an IPython kernel.

Procedure

1. Create a conda environment.

   Run the command below on the notebook terminal. In the command, **my-env** indicates a customizable virtual environment name. For details about conda parameters, see the **conda official website**.

   ```
   conda create --quiet --yes -n my-env python=3.6.5
   ```

   After the creation, run the **conda info --envs** command to view existing virtual environments, where you can see the **my-env** virtual environment.

   ```
   sh-4.4$conda info --envs
   # conda environments:
   #
   base                  *  /home/ma-user/anaconda3
   TensorFlow-2.1           /home/ma-user/anaconda3/envs/TensorFlow-2.1
   my-env                   /home/ma-user/anaconda3/envs/my-env
   python-3.7.10            /home/ma-user/anaconda3/envs/python-3.7.10                  /opt/conda/envs/my-
   env
   ```

2. Access the conda environment.

   ```
   source /home/ma-user/anaconda3/bin/activate /home/ma-user/anaconda3/envs/my-env
   ```

3. Install the following dependency packages in **my env**:

   ```
   pip install ipykernel
   ```

   If a version conflict occurs, you are advised to fix the following versions:

   ```
   pip install jupyter_core==5.3.0
   pip install jupyter_client==8.2.0
   pip install ipython==8.10.0
   pip install ipykernel==6.23.1
   ```

4. Add the virtual environment as an IPython kernel.

   The value of **--name** can be customized.

   ```
   python3 -m ipykernel install --user --name "my-py3-tensorflow-env"
   ```

   After the command is executed, the following information is displayed:

   ```
   (my-env) sh-4.4$python3 -m ipykernel install --user --name "my-py3-tensorflow-env"
   Installed kernelspec my-py3-tensorflow-env in /home/ma-user/.local/share/jupyter/kernels/my-py3-
   tensorflow-env
   ```

5. Customize the environment variables of the virtual environment kernel.

   Run the **cat /home/ma-user/.local/share/jupyter/kernels/my-py3-tensorflow-env/kernel.json** command to view the default configuration.

   ```
   {
    "argv": [
    "/home/ma-user/anaconda3/envs/my-env/bin/python3",
   ```



```
 "-m",
 "ipykernel_launcher",
 "-f",
 "{connection_file}"
],
"display_name": "my-py3-tensorflow-env",
"language": "python"
}
```

Add content in the **env** field as needed. For reference, see the following configuration. **PATH** indicates the path to the Python package of the virtual environment.

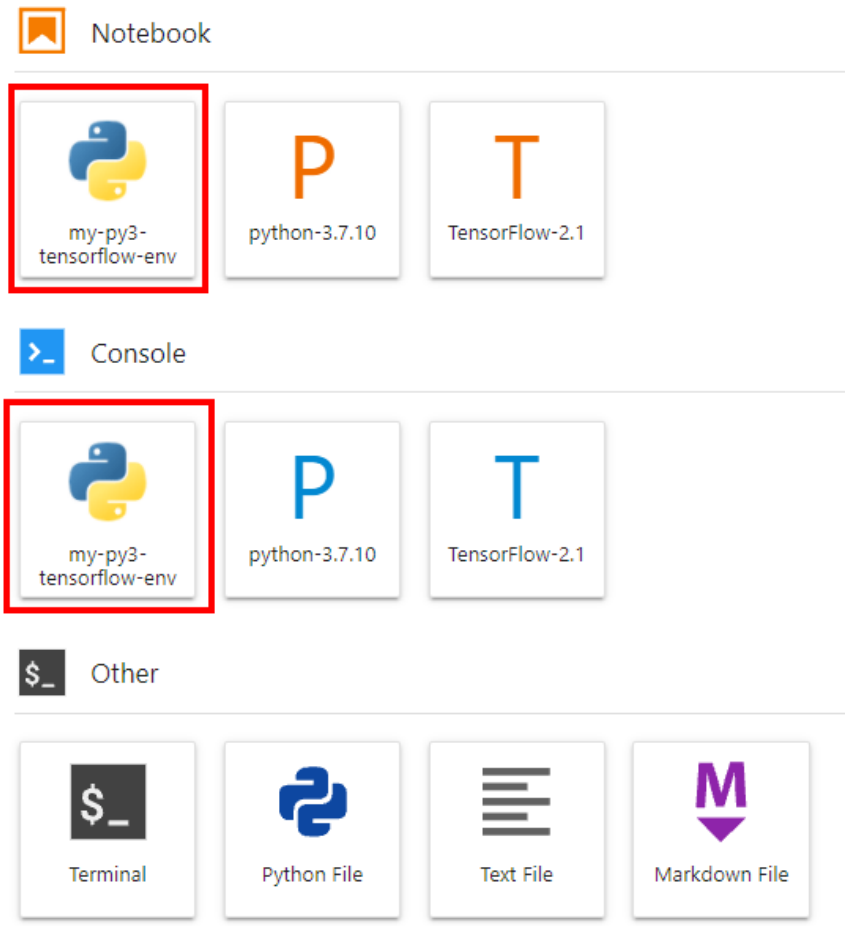
```
{
 "argv": [
  "/home/ma-user/anaconda3/envs/my-env/bin/python3",
  "-m",
  "ipykernel_launcher",
  "-f",
  "{connection_file}"
 ],
 "display_name": "my-py3-tensorflow-env",
 "language": "python",
 "env": {
      "PATH": "/home/ma-user/anaconda3/envs/my-env/bin:/opt/conda/bin:/usr/local/sbin:/usr/local/
bin:/usr/sbin:/usr/bin:/sbin:/bin:/home/ma-user/modelarts/ma-cli/bin",
      "http_proxy": "http://proxy-notebook.modelarts-dev-proxy.com:8083",
      "https_proxy": "http://proxy-notebook.modelarts-dev-proxy.com:8083",
      "ftp_proxy": "http://proxy-notebook.modelarts-dev-proxy.com:8083",
      "HTTP_PROXY": "http://proxy-notebook.modelarts-dev-proxy.com:8083",
      "HTTPS_PROXY": "http://proxy-notebook.modelarts-dev-proxy.com:8083",
      "FTP_PROXY": "http://proxy-notebook.modelarts-dev-proxy.com:8083"
 }
}
```

6. Manually delete invalid logos.

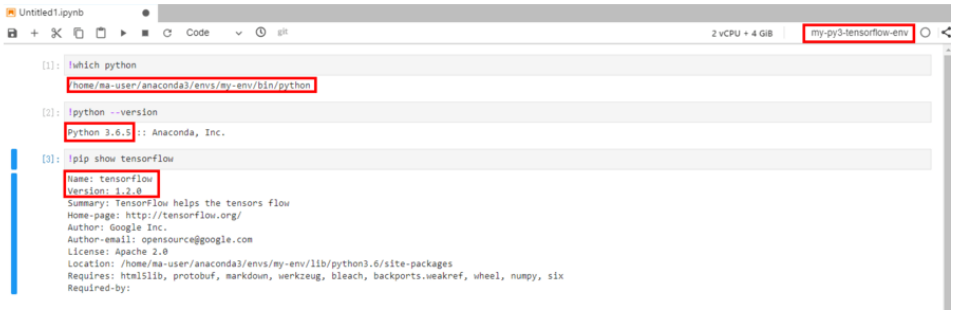
```
rm -r /home/ma-user/.local/share/jupyter/kernels/my-py3-tensorflow-env/logo-*
```

7. Access the IPython kernel of the virtual environment.

   Refresh the JupyterLab page. The custom virtual environment kernel is displayed.



Click **my-py3-tensorflow-env** to verify that the current environment is used.



8. Clear the environment.

   Delete the IPython kernel of the virtual environment.

   ```
   jupyter kernelspec uninstall my-py3-tensorflow-env
   ```

   Delete the virtual environment.

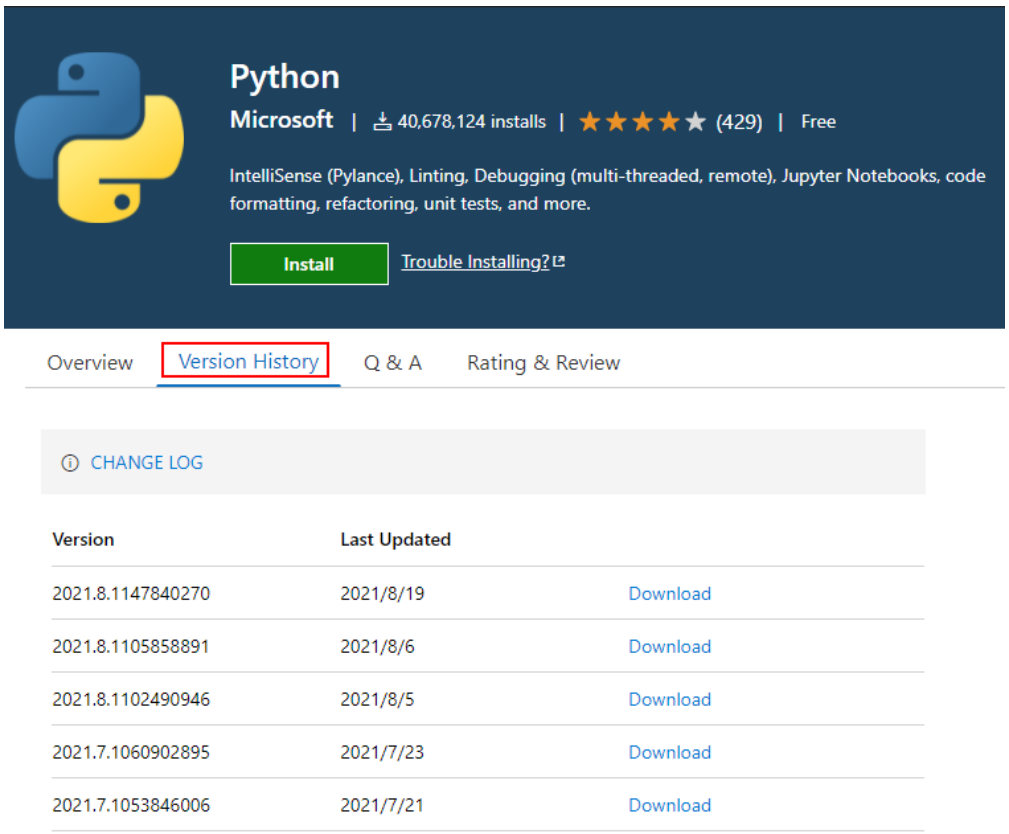
   ```
   conda env remove -n my-env
   ```

5.13 What Should I Do If It Is Unstable to Install the Remote Plug-in on a ModelArts Notebook Instance?

Method 1 (recommended): Using an offline package

1. Log in at the **official VS Code website** and search for the target Python plug-in.
2. Click the **Version History** tab of the plug-in and download the offline installation package.

Figure 5-6 Offline installation package of the Python plug-in



3. In local VS Code, drag the downloaded VSIX file to the remote notebook.
4. Right-click the file and choose **Install Extension VSIX** from the shortcut menu.

Method 2: Setting the default remote plug-in

Set the default remote plug-in in VS Code by following the instructions provided in **How Do I Configure the Default Plug-in Remotely Installed for VS Code on a ModelArts Notebook Instance?** This enables automatic plug-in installation when the plug-in is connected.

Method 3: Taking measures provided at **official VS Code website**

Tips (adjust parameter settings as needed):

```
"remote.SSH.connectTimeout": 10,
"remote.SSH.maxReconnectionAttempts": null,
"remote.downloadExtensionsLocally": true,
"remote.SSH.useLocalServer": false,
"remote.SSH.localServerDownload": "always",
```


5.14 How Do I Connect to a Restarted ModelArts Notebook Instance?

To resolve this issue, set notebook parameters **StrictHostKeyChecking no** and **UserKnownHostsFile=/dev/null** in the local **ssh config** file.

```
Host roma-local-cpu
   HostName x.x.x.x # IP address
   Port 22522
   User ma-user
   IdentityFile C:/Users/my.pem
   StrictHostKeyChecking no
   ForwardAgent yes
```

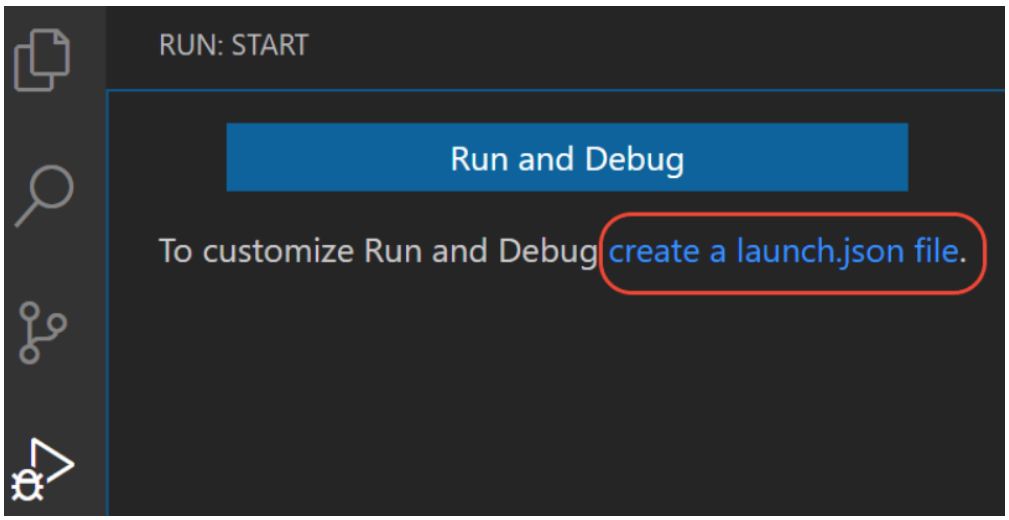
Note: SSH logins are insecure because the **known_hosts** file will be ignored during the logins.

5.15 What Should I Do If the Source Code Cannot Be Accessed When I Use VS Code to Debug Code on a ModelArts Notebook Instance?

If the **launch.json** file already exists, go to step 3.

Step 1: Open launch.json.

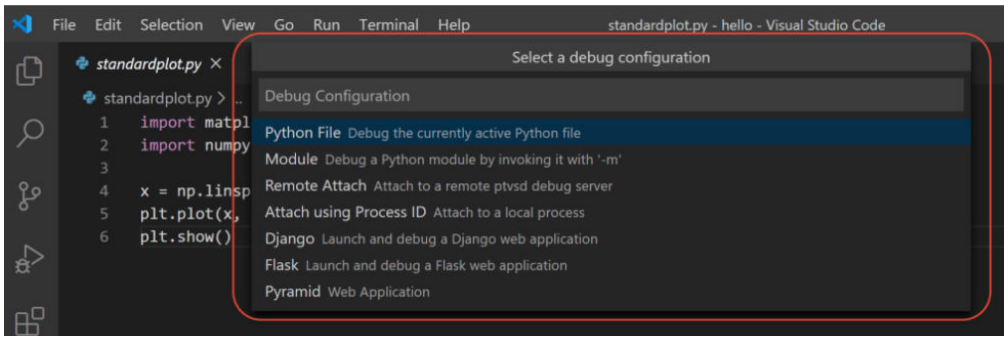
- Method 1: Click **Run** (**Ctrl+Shift+D**) in the menu bar on the left and click **create a launch.json file**.



- Method 2: In the menu bar, choose **Run** > **Open configurations**.

Step 2: Select a language.

To set a Python language, select **Python File** in **Select a debug configuration**. The operations for setting other languages are similar.



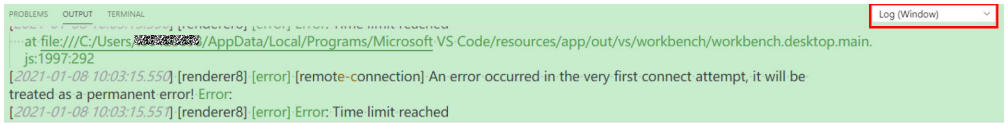
Step 3: Set justMyCode to false in launch.json.

```
{
    // Use IntelliSense to learn about possible attributes.
    // Hover to view descriptions of existing attributes.
    // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Python: Current file",
            "type": "python",
            "request": "launch",
            "program": "${file}",
            "console": "integratedTerminal",
            "justMyCode": false
        }
    ]
}
```

5.16 How Do I View Remote Logs Using VS Code on a ModelArts Notebook Instance?

1. In VS Code, press **Ctrl+Shift+P**.
2. Search for **show logs**.
3. Choose **Remote Server**.

Alternatively, switch logs in the red box shown in the following figure.



5.17 How Do I Open the VS Code Configuration File settings.json on a ModelArts Notebook Instance?

1. In VS Code, press **Ctrl+Shift+P**.
2. Search for **Open User Settings (JSON)**.

5.18 How Do I Set the Background Color of VS Code to Bean Green on a ModelArts Notebook Instance?

Add the following settings to the VS Code configuration file **settings.json**:

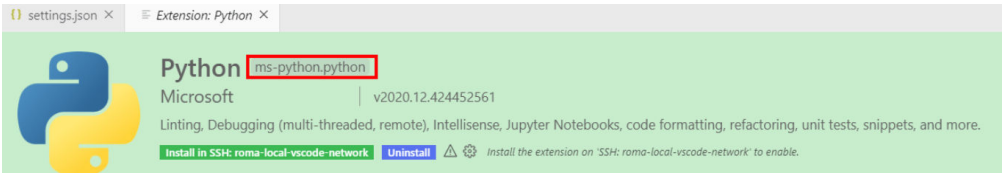
```
"workbench.colorTheme": "Atom One Light",
"workbench.colorCustomizations": {
"[Atom One Light]": {
 "editor.background": "#C7EDCC",
"sideBar.background": "#e7f0e7",
"activityBar.background": "#C7EDCC",
    },
},
```

5.19 How Do I Configure the Default Plug-in Remotely Installed for VS Code on a ModelArts Notebook Instance?

Add **remote.SSH.defaultExtensions**, for example, for automatically installing Python and Maven plug-ins, to the VS Code configuration file **settings.json**.

```
"remote.SSH.defaultExtensions": [
    "ms-python.python",
    "vscjava.vscode-maven"
 ],
```

To obtain a plug-in name, click the plug-in in VS Code.



5.20 How Do I Install a Local Plug-in Remotely or a Remote Plug-in Locally in ModelArts VS Code?

1. In VS Code, press **Ctrl+Shift+P**.
2. Search for **install local** and select the plug-in as required.

5.21 How Do I Use Multiple Ascend Cards for Debugging on a ModelArts Notebook Instance?

An Ascend multi-card training job runs in multi-process, multi-card mode. The number of cards is equal to the number of Python processes. The Ascend underlayer reads the environment variable **RANK_TABLE_FILE**, which has been configured in the development environment, without requiring manual configuration. For example, to run a job on eight cards, the code is as follows:

```
export RANK_SIZE=8
current_exec_path=$(pwd)
```



```
echo 'start training'
for((i=0;i<=$RANK_SIZE-1;i++));
do
echo 'start rank '$i
mkdir ${current_exec_path}/device$i
cd ${current_exec_path}/device$i
echo $i
export RANK_ID=$i
dev=`expr $i + 0`
echo $dev
export DEVICE_ID=$dev
python train.py > train.log 2>&1 &
done
```

Set the environment variable **DEVICE_ID** in **train.py**.

```
devid = int(os.getenv('DEVICE_ID'))
context.set_context(mode=context.GRAPH_MODE, device_target="Ascend", device_id=devid)
```

5.22 Why Are the Training Speeds Similar When Different Resource Flavors Are Used for Training on ModelArts Notebook Instances?

If your training job is single-process in code, the training speed is basically the same no matter when the notebook flavor of 8 vCPUs and 64 GB of memory or the flavor of 72 vCPUs and 512 GB of memory is used. For example, if your training job uses 2 vCPUs and 4 GB of memory, the training speed is similar no matter when you use the notebook flavor of 4 vCPUs and 8 GB of memory or the flavor of 8 vCPUs and 64 GB of memory.

If your training job is multi-process in code, the training speed backed by the notebook flavor of 72 vCPUs and 512 GB of memory is higher than that backed by the notebook flavor of 8 vCPUs and 64 GB of memory.

5.23 How Do I Perform Incremental Training When Using MoXing on a ModelArts Notebook Instance?

If you are not satisfied with previous training results when using MoXing to build a model, you can perform incremental training after modifying some data and label information.

Adding Incremental Training Parameters to mox.run

After modifying labeling data or datasets, you can modify the **log_dir** parameter in and add the **checkpoint_path** parameter to **mox.run**. Set **log_dir** to a new directory and **checkpoint_path** to the output path of the previous training results. If the output path is an OBS directory, set the path to a value starting with **obs://**.

If labels are changed for label data, perform operations in **If Labels Are Changed** before running **mox.run**.

```
mox.run(input_fn=input_fn,
        model_fn=model_fn,
        optimizer_fn=optimizer_fn,
```



```
    run_mode=flags.run_mode,
    inter_mode=mox.ModeKeys.EVAL if use_eval_data else None,
    log_dir=log_dir,
    batch_size=batch_size_per_device,
    auto_batch=False,
    max_number_of_steps=max_number_of_steps,
    log_every_n_steps=flags.log_every_n_steps,
    save_summary_steps=save_summary_steps,
    save_model_secs=save_model_secs,
    checkpoint_path=flags.checkpoint_url,
    export_model=mox.ExportKeys.TF_SERVING)
```

If Labels Are Changed

If the labels in a dataset have changed, execute the following statement. The statement must be executed before running **mox.run**.

In the statement, the **logits** variable indicates classification layer weights in different networks, and different parameters are configured. Set this parameter to the corresponding keyword.

```
mox.set_flag('checkpoint_exclude_patterns', 'logits')
```

If the built-in network of MoXing is used, the corresponding keyword needs to be obtained by calling the following API. In this example, the **Resnet_v1_50** keyword is the value of **logits**.

```
import moxing.tensorflow as mox

model_meta = mox.get_model_meta(mox.NetworkKeys.RESNET_V1_50)
logits_pattern = model_meta.default_logits_pattern
print(logits_pattern)
```

You can also obtain a list of networks supported by MoXing by calling the following API:

```
import moxing.tensorflow as mox
print(help(mox.NetworkKeys))
```

The following information is displayed:

```
Help on class NetworkKeys in module
moxing.tensorflow.nets.nets_factory:

class NetworkKeys(builtins.object)
 |  Data descriptors defined here:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
 |
 |  ----------------------------------------------------------------------
 |  Data and other attributes defined here:
 |
 |  ALEXNET_V2 = 'alexnet_v2'
 |
 |  CIFARNET = 'cifarnet'
 |
 |  INCEPTION_RESNET_V2 = 'inception_resnet_v2'
 |
 |  INCEPTION_V1 = 'inception_v1'
 |
 |  INCEPTION_V2 = 'inception_v2'
 |
 |  INCEPTION_V3 = 'inception_v3'
```



```
INCEPTION_V4 = 'inception_v4'
LENET = 'lenet'
MOBILENET_V1 = 'mobilenet_v1'
MOBILENET_V1_025 = 'mobilenet_v1_025'
MOBILENET_V1_050 = 'mobilenet_v1_050'
MOBILENET_V1_075 = 'mobilenet_v1_075'
MOBILENET_V2 = 'mobilenet_v2'
MOBILENET_V2_035 = 'mobilenet_v2_035'
MOBILENET_V2_140 = 'mobilenet_v2_140'
NASNET_CIFAR = 'nasnet_cifar'
NASNET_LARGE = 'nasnet_large'
NASNET_MOBILE = 'nasnet_mobile'
OVERFEAT = 'overfeat'
PNASNET_LARGE = 'pnasnet_large'
PNASNET_MOBILE = 'pnasnet_mobile'
PVANET = 'pvanet'
RESNET_V1_101 = 'resnet_v1_101'
RESNET_V1_110 = 'resnet_v1_110'
RESNET_V1_152 = 'resnet_v1_152'
RESNET_V1_18 = 'resnet_v1_18'
RESNET_V1_20 = 'resnet_v1_20'
RESNET_V1_200 = 'resnet_v1_200'
RESNET_V1_50 = 'resnet_v1_50'
RESNET_V1_50_8K = 'resnet_v1_50_8k'
RESNET_V1_50_MOX = 'resnet_v1_50_mox'
RESNET_V1_50_OCT = 'resnet_v1_50_oct'
RESNET_V2_101 = 'resnet_v2_101'
RESNET_V2_152 = 'resnet_v2_152'
RESNET_V2_200 = 'resnet_v2_200'
RESNET_V2_50 = 'resnet_v2_50'
RESNEXT_B_101 = 'resnext_b_101'
RESNEXT_B_50 = 'resnext_b_50'
RESNEXT_C_101 = 'resnext_c_101'
RESNEXT_C_50 = 'resnext_c_50'
```



```
| VGG_16 = 'vgg_16'
|
| VGG_16_BN = 'vgg_16_bn'
|
| VGG_19 = 'vgg_19'
|
| VGG_19_BN = 'vgg_19_bn'
|
| VGG_A = 'vgg_a'
|
| VGG_A_BN = 'vgg_a_bn'
|
| XCEPTION_41 = 'xception_41'
|
| XCEPTION_65 = 'xception_65'
|
| XCEPTION_71 = 'xception_71'
```

5.24 How Do I View the GPU Usage on a ModelArts Notebook Instance?

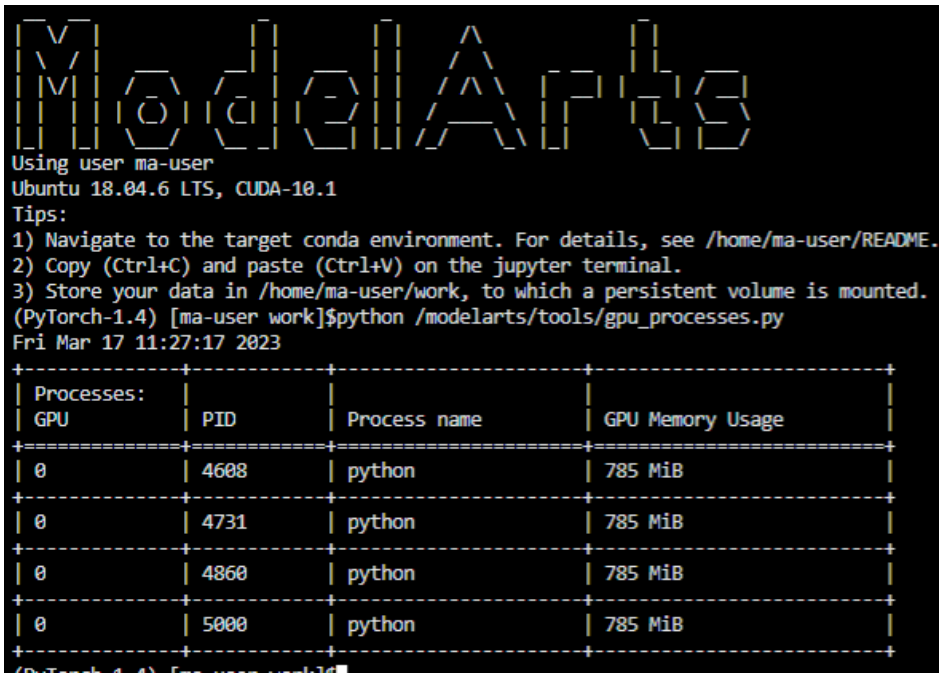
If you select GPU when creating a notebook instance, perform the following operations to view GPU usage:

1. Log in to the ModelArts management console, and choose **Development Workspace > Notebook**.
2. In the **Operation** column of the target notebook instance in the notebook list, click **Open** to go to the **Jupyter** page.
3. On the **Files** tab page of the **Jupyter** page, click **New** and select **Terminal**. The **Terminal** page is displayed.
4. Run the following command to view GPU usage:
   ```
   nvidia-smi
   ```
5. Check which processes in the current notebook instance use GPUs.

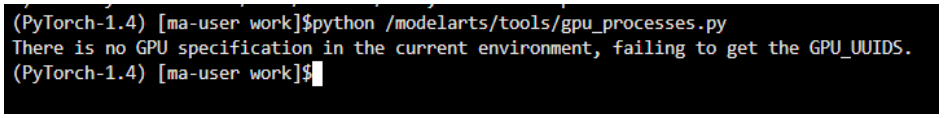
   Method 1:

   ```
   python /modelarts/tools/gpu_processes.py
   ```

   The following figure shows the case that the current process is using GPUs.



The following figure shows the case that the current process is not using GPUs.



Method 2:

Open **/resource_info/gpu_usage.json** and view the processes that are using GPUs.

```
{
    <notebook name>: {
        <GPU0 UUID>: [
            {
                "pid": 2263,
                "processName": "python",
                "gpuMemoryUsage": "4935Mi"
            },
            {...}
        ]
        <GPU1 UUID>: [...]
    }
}
```

If no process is using GPUs, the file may be unavailable or empty.

5.25 How Can I Print the GPU Usage in Code on a ModelArts Notebook Instance?

Run the shell or python command to obtain the GPU usage.

Using the shell Command

1. Run the **nvidia-smi** command.

   This operation relies on CUDA NVCC.

   ```
   watch -n 1 nvidia-smi
   ```

   ```
   Every 1.0s: nvidia-smi

   Mon Oct 25 15:20:11 2021
   +-----------------------------------------------------------------------------+
   | NVIDIA-SMI 440.33.01    Driver Version: 440.33.01    CUDA Version: 10.2     |
   |-------------------------------+----------------------+----------------------+
   | GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
   | Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
   |===============================+======================+======================|
   |   0  Tesla V100-SXM2...  On   | 00000000:5F:00.0 Off |                    0 |
   | N/A   31C    P0    43W / 300W |      0MiB / 32510MiB |      0%      Default |
   +-------------------------------+----------------------+----------------------+
   |   1  Tesla V100-SXM2...  On   | 00000000:B5:00.0 Off |                    0 |
   | N/A   34C    P0    44W / 300W |      0MiB / 32510MiB |      0%      Default |
   +-------------------------------+----------------------+----------------------+

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

2. Run the **gpustat** command.

   ```
   pip install gpustat
   gpustat -cp -i
   ```

   ```
   notebook-6a654129-698e-4635-b6be-67aedbdd4c54  Mon Oct 25 15:19:11 2021  440.33.01
   [0] Tesla V100-SXM2-32GB | 31'C,   0 % |     0 / 32510 MB |
   [1] Tesla V100-SXM2-32GB | 34'C,   0 % |     0 / 32510 MB |
   ```

   To stop the command execution, press **Ctrl+C**.

Using the python Command

1. Run the **nvidia-ml-py3** command (commonly used).

   ```
   !pip install nvidia-ml-py3
   import nvidia_smi
   nvidia_smi.nvmlInit()
   deviceCount = nvidia_smi.nvmlDeviceGetCount()
   for i in range(deviceCount):
       handle = nvidia_smi.nvmlDeviceGetHandleByIndex(i)
       util = nvidia_smi.nvmlDeviceGetUtilizationRates(handle)
       mem = nvidia_smi.nvmlDeviceGetMemoryInfo(handle)
       print(f"|Device {i}| Mem Free: {mem.free/1024**2:5.2f}MB / {mem.total/1024**2:5.2f}MB | gpu-util: {util.gpu:3.1%} | gpu-mem: {util.memory:3.1%} |")
   ```

   ```
   Output:
   |Device 0| Mem Free: 32510.44MB / 32510.50MB | gpu-util: 0.0% | gpu-mem: 0.0% |
   |Device 1| Mem Free: 32510.44MB / 32510.50MB | gpu-util: 0.0% | gpu-mem: 0.0% |
   ```

2. Run the **nvidia_smi**, **wapper**, and **prettytable** commands.

   Use the decorator to obtain the GPU usage in real time during model training.

   ```
   def gputil_decorator(func):
       def wrapper(*args, **kwargs):
           import nvidia_smi
           import prettytable as pt

           try:
               table = pt.PrettyTable(['Devices','Mem Free','GPU-util','GPU-mem'])
               nvidia_smi.nvmlInit()
   ```



```
        deviceCount = nvidia_smi.nvmlDeviceGetCount()
        for i in range(deviceCount):
            handle = nvidia_smi.nvmlDeviceGetHandleByIndex(i)
            res = nvidia_smi.nvmlDeviceGetUtilizationRates(handle)
            mem = nvidia_smi.nvmlDeviceGetMemoryInfo(handle)
            table.add_row([i, f"{mem.free/1024**2:5.2f}MB/{mem.total/1024**2:5.2f}MB",
f"{res.gpu:3.1%}", f"{res.memory:3.1%}"])

    except nvidia_smi.NVMLError as error:
        print(error)

    print(table)
    return func(*args, **kwargs)
  return wrapper
```

```
Output:

+---------+-----------------------+----------+----------+
| Devices |        Mem Free       | GPU-util | GPU-mem |
+---------+-----------------------+----------+----------+
|    0    | 32510.44MB/32510.50MB |   0.0%   |   0.0%  |
|    1    | 32510.44MB/32510.50MB |   0.0%   |   0.0%  |
+---------+-----------------------+----------+----------+
```

3. Run the **pynvml** command.

   Run **nvidia-ml-py3** to directly obtain the nvml c-lib library, without using **nvidia-smi**. Therefore, this command is recommended.

```
from pynvml import *
nvmlInit()
handle = nvmlDeviceGetHandleByIndex(0)
info = nvmlDeviceGetMemoryInfo(handle)
print("Total memory:", info.total)
print("Free memory:", info.free)
print("Used memory:", info.used)
```

```
Output:
Total memory: 34089730048
Free memory: 34089664512
Used memory: 65536
```

4. Run the **gputil** command.

```
!pip install gputil
import GPUtil as GPU
GPU.showUtilization()
```

```
Output:

| ID | GPU | MEM |
------------------
|  0 |  0% | 25% |
|  1 |  0% |  0% |
```

```
import GPUtil as GPU
GPUs = GPU.getGPUs()
for gpu in GPUs:
    print("GPU RAM Free: {0:.0f}MB | Used: {1:.0f}MB | Util {2:3.0f}% | Total
{3:.0f}MB".format(gpu.memoryFree, gpu.memoryUsed, gpu.memoryUtil*100, gpu.memoryTotal))
```

```
Output:
GPU RAM Free: 32510MB | Used: 0MB | Util   0% | Total 32510MB
GPU RAM Free: 32510MB | Used: 0MB | Util   0% | Total 32510MB
```


When using a deep learning framework such as PyTorch or TensorFlow, you can also use the APIs provided by the framework for query.

5.26 What Are the Relationships Among JupyterLab Directories, Terminal Files, and OBS Files on ModelArts Notebook Instances?

- Files stored in JupyterLab are the same as those in the work directory on the **Terminal** page. That is, the files are created on your notebook instances or synchronized from OBS.
- Notebook instances with OBS storage mounted can synchronize files from OBS to JupyterLab using the JupyterLab upload and download functions. The files on the **Terminal** page are the same as those in JupyterLab.
- Notebook instances with EVS storage mounted can read files from OBS to JupyterLab using the MoXing API or SDKs. The files on the **Terminal** page are the same as those in JupyterLab.

5.27 How Do I Use ModelArts Datasets on a ModelArts Notebook Instance?

Datasets created on ModelArts are stored in OBS. To use these datasets on a notebook instance, download them from OBS to the notebook instance.

For details, see **How Do I Upload a File from a Notebook Instance to OBS or Download a File from OBS to a Notebook Instance in ModelArts?**

5.28 pip and Common Commands

pip is a common Python package management tool. It allows you to search for, download, install, and uninstall Python packages.

Common pip commands:

```
pip --help # Obtain help information.
pip install SomePackage==XXXX # Install a specified version.
pip install SomePackage # Install the latest version.
pip uninstall SomePackage # Uninstall a software version.
```

For other commands, run the **pip --help** command.

5.29 What Are the Sizes of the /cache Directories for Resources with Varying Specifications on ModelArts Notebook Instances?

When creating a notebook instance, you can select resources based on the data volume. ModelArts mounts disks to **/cache**. You can use this directory to store temporary files. The **/cache** directory shares resources with the code directory. The directory size varies depending on resource specifications.

- Old resource pool: Local disks are used. For details about the size of the **/cache** directory, see **Table 5-1**. The mapping format is as follows:
  - Currently, the **/cache** directory cannot be configured for CPUs.
  - If you use a single GP or Ascend processor, the /cache directory size is limited to 500 GB.
  - If you use more than one processing unit (PU), the **/cache** directory size depends on the number of PUs. The size is the number of PUs times 500 GB, with a maximum of 3 TB.
- New resource pool: A cluster is created and EVS disk mounting is enabled. When you create a dedicated resource pool, the set disk capacity is the total available for allocation. Each PU gets a share of this disk space based on the total number of PUs in the pool.

  The **/cache** directory size for an instance is calculated as: Number of PUs used by the instance × **/Cache** directory size per PU.

Table 5-1 /cache directory sizes for different notebook specifications

Specification	/cache Directory Size
GP x 0.25	500 GB x 0.25
GP x 0.5	500 GB x 0.5
GP x 1	500 GB
GP x 2	500 GB x 2
GP x 4	500 GB x 4
GP x 8	3 TB
Huawei Cloud AI processor x 1	500 GB
Huawei Cloud AI processor x 2	500 GB x 2
Huawei Cloud AI processor x 4	500 GB x 4
Huawei Cloud AI processor x 8	3 TB
CPU	N/A

5.30 What Is the Impact of Resource Overcommitment on ModelArts Notebook Instances?

Notebook overcommitment refers to the sharing of GPUs and memory within a node. To fully utilize resources, they are overcommitted in dedicated pools.

Example: A dedicated pool has one CPU node with 8 vCPUs and 64 GB of memory. If you create a notebook instance with 2 vCPUs and 8 GB of memory, a maximum of 6.67 notebook instances (8 vCPUs/(2 vCPUs x 0.6)) can be started due to overcommitment with an overcommitment ratio of 0.6. In this case, at least 1.2 vCPUs are required for starting the notebook instance, and a maximum of 2 vCPUs

are used for running the notebook instance. Similarly, at least 4.8 GB memory is required, and a maximum of 8 GB memory is used for running the notebook instance.

Instances may be forcibly terminated due to overcommitment. For example, if six instances with 2 vCPUs are started on an 8 vCPUs node and the CPU usage of one instance exceeds the upper limit (8 vCPUs) of the node, Kubernetes forcibly terminates the instance that uses the most resources.

Do not overcommit resources as it may result in instance restart.

5.31 How Do I Install External Libraries in a Notebook Instance?

Multiple environments such as Jupyter and Python have been integrated into ModelArts notebook to support many frameworks, including TensorFlow, MindSpore, PyTorch, and Spark. You can use **pip install** to install external libraries in Jupyter Notebook or on the **Terminal** page.

Installing External Libraries in Jupyter Notebook

You can use JupyterLab to install Shapely in the **TensorFlow-1.8** environment.

1. Open a notebook instance and access the **Launcher** page.
2. In the **Notebook** area, click **TensorFlow-1.8** and create an IPYNB file.
3. In the new notebook instance, enter the following command in the code input bar:

   !pip install Shapely

Installing External Libraries on the Terminal Page

You can use **pip** to install external libraries in the **TensorFlow-1.8** environment on the **Terminal** page. For example, to install Shapely:

1. Open a notebook instance and access the **Launcher** page.
2. In the **Other** area, click **Terminal** and create a terminal file.
3. Enter the following commands in the code input box to obtain the kernel of the current environment and activate the Python environment on which the installation depends:

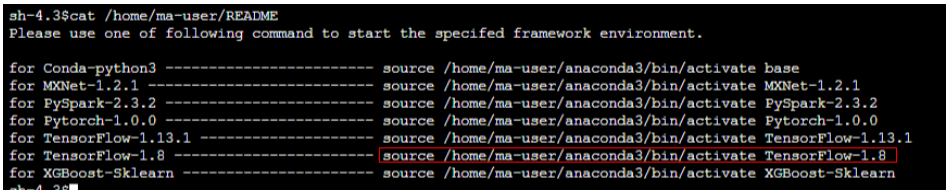
   cat /home/ma-user/README

   source /home/ma-user/anaconda3/bin/activate TensorFlow-1.8

   NOTE

   To install TensorFlow in another Python environment, replace **TensorFlow-1.8** in the command with the target engine.

Figure 5-7 Activating the environment



4. Run the following command in the code input box to install Shapely:

   pip install Shapely

5.32 How Do I Handle Unstable Internet Access Speed in ModelArts Notebook?

ModelArts provides a free, shared network proxy service for accessing external resources when using notebook instances, facilitating development by enabling convenient downloads.

However, because this network proxy is free and shared, its performance can be significantly affected by real-time usage. When many users download resources simultaneously, network bandwidth can become congested, leading to slower download speeds. Conversely, when fewer users are accessing the proxy, download speeds may improve. ModelArts cannot guarantee stable and fast download speeds at all times.

To avoid issues caused by unstable network downloads, schedule downloads during off-peak hours. Additionally, plan ahead and consider alternative solutions if high download speed is essential.

5.33 Can I Use GDB in a Notebook Instance?

No. GDB needs Docker with privileged containers. For security, the development environment does not allow privileged containers. GDB cannot be used in notebook instances.

6 ModelArts Standard Model Training

6.1 What Should I Do If the Model Trained in ModelArts Is Underfitting?

1. Increasing model complexity
   - For an algorithm, add more high-order items to the regression model, improve the depth of the decision tree, or increase the number of hidden layers and hidden units of the neural network to increase model complexity.
   - Discard the original algorithm and use a more complex algorithm or model. For example, use a neural network to replace the linear regression, and use the random forest to replace the decision tree.
2. Adding more features to make input data more expressive
   - Feature mining is critical. In particular, a small set of highly expressive features can often be more effective than a larger set of less expressive ones.
   - Feature quality is the focus.
   - To explore highly expressive features, you must have an in-depth understanding of data and application scenarios, which depends on experience.
3. Adjusting parameters and hyperparameters
   - Neural network: learning rate, learning attenuation rate, number of hidden layers, number of units in a hidden layer, β1 and β2 parameters in the Adam optimization algorithm, and batch_size
   - Other algorithms: number of trees in the random forest, number of clusters in *k*-means, and regularization parameter λ
4. Adding training data, which is of little effect

   Underfitting is usually caused by weak model learning capabilities. Adding data cannot significantly increase the training performance.
5. Reducing regularization constraints

Regularization aims to prevent model overfitting. If a model is underfitting instead of overfitting, reduce the regularization parameter **λ** or directly remove the regularization item.

6.2 How Do I Obtain a Trained Model in ModelArts?

Models trained using a custom or subscription algorithm are stored in specified OBS paths for you to download.

6.3 How Do I Obtain RANK_TABLE_FILE for Distributed Training in ModelArts?

ModelArts automatically provides the **RANK_TABLE_FILE** file for you. Obtain the file location through environment variables.

- Open the notebook terminal and run the following command to view **RANK_TABLE_FILE**:
  ```
  env | grep RANK
  ```
- In a training job, add the following code to the first line of the training startup script to print the value of **RANK_TABLE_FILE**:
  ```
  os.system('env | grep RANK')
  ```

6.4 How Do I Configure Input and Output Data for Model Training in ModelArts?

ModelArts allows you to upload a custom algorithm for creating training jobs. **Create an algorithm** and upload it to an OBS bucket. For details about how to create an algorithm, see **Developing Code for Training Using a Preset Image**. For details about how to create a training job, see **Creating a Training Job**.

Parsing Input and Output Paths

When a ModelArts model reads data stored in OBS or outputs data to a specified OBS path, perform the following operations to configure the input and output data:

1. Parse the input and output paths in the training code. The following method is recommended:
   ```
   import argparse
   # Create a parsing task.
   parser = argparse.ArgumentParser(description="train mnist",
                                    formatter_class=argparse.ArgumentDefaultsHelpFormatter)
   # Add parameters.
   parser.add_argument('--train_url', type=str,
                       help='the path model saved')
   parser.add_argument('--data_url', type=str, help='the training data')
   # Parse the parameters.
   args, unknown = parser.parse_known_args()
   ```
   After the parameters are parsed, use **data_url** and **train_url** to replace the paths to the data source and the data output, respectively.
2. When using a preset image to create an algorithm, set the defined input and output parameters based on the code parameters in **1**.

- Training data is a must for algorithm development. You are advised to set the input parameter name to **data_url**, indicating the input data source. You can also customize code parameters based on the algorithm code in **1**.
- After model training is complete, the trained model and the output information must be stored in an OBS path. By default, **Output** specifies the model output and the code path parameter is **train_url**. You can also customize the output path parameters based on the algorithm code in **1**.

3. When creating a training job, configure the input and output paths.

   Select an OBS path or dataset path as the training input, and an OBS path for the output.

6.5 How Do I Improve Training Efficiency While Reducing Interaction with OBS in ModelArts?

Scenarios

When you use ModelArts for custom deep learning training, training data is typically stored in OBS. If the volume of training data is large (for example, greater than 200 GB), a GP resource pool is required, and the training efficiency is low.

To improve training efficiency while reducing interaction with OBS, perform the following operations for optimization.

Optimization Principles

For the GP resource pool provided by ModelArts, 500 GB NVMe SSDs are attached to each training node for free. The SSDs are attached to the **/cache** directory. The lifecycle of data in the **/cache** directory is the same as that of a training job. After the training job is complete, all content in the **/cache** directory is cleared to release space for the next training job. Therefore, you can copy data from OBS to the **/cache** directory during training so that data can be read from the **/cache** directory until the training is finished. After the training is complete, content in the **/cache** directory will be automatically cleared.

Optimization Methods

TensorFlow code is used as an example.

The following is code before optimization:

```
...
tf.flags.DEFINE_string('data_url', '', 'dataset directory.')
FLAGS = tf.flags.FLAGS
mnist = input_data.read_data_sets(FLAGS.data_url, one_hot=True)
```

The following is an example of the optimized code. Data is copied to the **/cache** directory.

```
...
tf.flags.DEFINE_string('data_url', '', 'dataset directory.')
FLAGS = tf.flags.FLAGS
import moxing as mox
```



```
TMP_CACHE_PATH = '/cache/data'
mox.file.copy_parallel('FLAGS.data_url', TMP_CACHE_PATH)
mnist = input_data.read_data_sets(TMP_CACHE_PATH, one_hot=True)
```

6.6 How Do I Define Path Variables When Using MoXing to Copy Data in ModelArts?

Symptom

```
mox.file.copy_parallel(src_obs_dir=input_storage,'obs://dyyolov8/yolov5_test/yolov5-7.0/datasets'),
```

How do I define an OBS path as a variable in the **mox** function?

Solution

The following is an example of defining a variable:

```
input_storage = './test.py'
import moxing as mox
mox.file.copy_parallel(input_storage,'obs://dyyolov8/yolov5_test/yolov5-7.0/datasets')
```

6.7 How Do I Create a Training Job That References a Third-Party Dependency Package in ModelArts?

ModelArts allows you to install third-party dependency packages for model training. After the **pip-requirements.txt** file is stored in the training code directory, the system runs the command below to install the specified Python packages before the training boot file is executed.

```
pip install -r pip-requirements.txt
```

Only training jobs created using a preset image can reference dependency packages for model training.

NOTE

Any one of the following file names can be used. This section uses **pip-requirements.txt** as an example.

- pip-requirement.txt
- pip-requirements.txt
- requirement.txt
- requirements.txt

- For details about the code directory, see **Storing the Installation File in the Code Directory**.
- For details about the specifications of **pip-requirements.txt**, see **Installation File Specifications**.

Storing the Installation File in the Code Directory

- If you use **My algorithm** to create a training job, you can store related files in the configured **Code Directory** when creating an algorithm. The **Boot Mode** of the algorithm must be **Preset image**.

- If you use **Custom algorithm** to create a training job, you can store related files in the configured **Code Directory**. The **Boot Mode** must be **Preset image**.

Before creating a training job, upload related files to OBS. For details about the file packaging requirements, see **Installation File Specifications**.

Installation File Specifications

The installation file varies depending on the dependency package type.

- **Open-source installation packages**

  NOTE

  Installation using the source code from GitHub is not supported.

  Create a file named **pip-requirements.txt** in the code directory, and specify the name and version number of the dependency package in the file. The format is *[Package name]==[Version]*.

  Take for example, an OBS path specified by **Code Dir** that contains model files and the **pip-requirements.txt** file. The code directory structure would be as follows:

```
|---OBS path to the model boot file
    |---model.py              #Model boot file
    |---pip-requirements.txt   #Defined configuration file, which specifies the name and version of the dependency package
```

  The following shows the content of the **pip-requirements.txt** file:

```
alembic==0.8.6
bleach==1.4.3
click==6.6
```

- **WHL packages**

  If the training background does not support the download of open source installation packages or use of user-compiled WHL packages, the system cannot automatically download and install the package. In this case, place the WHL package in the code directory, create a file named **pip-requirements.txt**, and specify the name of the WHL package in the file. The dependency package must be a **.whl** file.

  Take for example, an OBS path specified by **Code Dir** that contains model files, the **.whl** file, and the **pip-requirements.txt** file. The code directory structure would be as follows:

```
|---OBS path to the model boot file
    |---model.py              #Model boot file
    |---XXX.whl               #Dependency package. If multiple dependencies are required, place multiple dependency packages here.
    |---pip-requirements.txt   #Defined configuration file, which specifies the name of the dependency package
```

  The following shows the content of the **pip-requirements.txt** file:

```
numpy-1.15.4-cp36-cp36m-manylinux1_x86_64.whl
tensorflow-1.8.0-cp36-cp36m-manylinux1_x86_64.whl
```


6.8 How Do I Install C++ Dependent Libraries During ModelArts Training?

A third-party library may be used during job training. The following uses C++ as an example to describe how to install a third-party library.

1. Download source code to a local PC and upload it to OBS. For details about how to upload a file using OBS Browser, see **Uploading a File**.
2. Use MoXing to copy the source code uploaded to OBS to a notebook instance in the development environment.

   The following is a code sample for copying data to a notebook instance in a development environment with EVS mounted:

   ```
   import moxing as mox
   mox.file.make_dirs('/home/ma-user/work/data')
   mox.file.copy_parallel('obs://bucket-name/data', '/home/ma-user/work/data')
   ```
3. On the **Files** tab page of the **Jupyter** page, click **New** and select **Terminal**. Run the following command to go to the target path, and check whether the source code has been downloaded, that is, whether the **data** file exists.

   ```
   cd /home/ma-user/work
   ls
   ```
4. Compile code in **Terminal** based on service requirements.
5. Use MoXing to copy the compilation results to OBS. The following is a code example.

   ```
   import moxing as mox
   mox.file.make_dirs('/home/ma-user/work/data')
   mox.file.copy_parallel('/home/ma-user/work/data', 'obs://bucket-name/file)
   ```
6. During training, use MoXing to copy the compilation result from OBS to the container. The following is a code example.

   ```
   import moxing as mox
   mox.file.make_dirs('/cache/data')
   mox.file.copy_parallel('obs://bucket-name/data', '/cache/data')
   ```

6.9 How Do I Check Whether a Folder Copy Is Complete During Job Training in ModelArts?

In the script for the training job boot file, run the following commands to obtain the sizes of the source folder and the copy. Then determine whether folder is copied based on the command output.

```
import moxing as mox
mox.file.get_size('obs://bucket_name/obs_file',recursive=True)
```

get_size indicates the size of the file or folder to be obtained. **recursive=True** indicates that the type is folder. **True** indicates that the type is folder, and **False** indicates that the type is file.

If the command output is consistent, the folder is copied. If the command output is inconsistent, the folder is not copied.

6.10 How Do I Load Some Well Trained Parameters During Job Training in ModelArts?

During job training, some parameters need to be loaded from a pre-trained model to initialize the current model. You can use the following methods to load the parameters:

1. View all parameters by using the following code:
   ```
   from moxing.tensorflow.utils.hyper_param_flags import mox_flags
   print(mox_flags.get_help())
   ```
2. Specify the parameters to be restored during model loading. **checkpoint_include_patterns** is the parameter that needs to be restored, and **checkpoint_exclude_patterns** is the parameter that does not need to be restored.
   ```
   checkpoint_include_patterns: Variables names patterns to include when restoring checkpoint. Such as: conv2d/weights.
   checkpoint_exclude_patterns: Variables names patterns to include when restoring checkpoint. Such as: conv2d/weights.
   ```
3. Specify a list of parameters to be trained. **trainable_include_patterns** is a list of parameters that need to be trained, and **trainable_exclude_patterns** is a list of parameters that do not need to be trained.
   ```
   --trainable_exclude_patterns: Variables names patterns to exclude for trainable variables. Such as: conv1,conv2.
   --trainable_include_patterns: Variables names patterns to include for trainable variables. Such as: logits.
   ```

6.11 What Should I Do If I Cannot Access the Folder Using os.system ('cd xxx') During Training in ModelArts?

If you cannot access the corresponding folder by using **os.system('cd xxx')** in the boot script of the training job, you are advised to use the following method:

```
import os
os.chdir('/home/work/user-job-dir/xxx')
```

6.12 How Do I Obtain the Dependency File Path from Training Code in ModelArts?

Since locally developed code must be uploaded to the ModelArts backend, you may set an invalid dependency file path. A recommended general solution to this problem is that you to use the OS API to obtain the absolute path of the dependency files.

The following shows an example of obtaining the path of dependency files in other folders using the OS API.

File directory structure:

```
project_root            #Root directory of code
 └─bootfile.py          #Boot file
```



```
└─otherfileDirectory       #Directory of dependency files
    └─otherfile.py       #Dependency files
```

Add the following code to the boot file to obtain the path (**otherfile_path**) of dependency files:

```
import os
current_path = os.path.dirname(os.path.realpath(__file__)) # Obtain the path of the boot file bootfile.py.
project_root = os.path.dirname(current_path) # Obtain the root directory of the project using the path of the boot file, which is the code directory set on ModelArts console.
otherfile_path = os.path.join(project_root, "otherfileDirectory", "otherfile.py")  # Obtain the path of the dependency files using the root directory of the project.
```

6.13 How Do I Obtain the Actual File Path in a Training Container in ModelArts?

To obtain the actual path to a file in a container, use Python.

```
os.getcwd() # Obtain the current work directory (absolute path) of the file.
os.path.realpath(__ file __) # Obtain the absolute path of the file.
```

You can also use other methods of obtaining a file path through the search engine and use the obtained path to read and write the file.

6.14 What Are the Sizes of the /cache Directories for Resources with Varying Specifications in Training Jobs in ModelArts?

When creating a training job, you can select resources based on the size of the training job.

ModelArts mounts a disk to **/cache**. You can use this directory to store temporary files. The **/cache** directory shares resources with the code directory. The directory has different capacities for different resource specifications.

NOTE

- The eviction policy of Kubernetes disks is 90%. Therefore, the effective size of a disk is 90% of the **cache** directory capacity.
- The local disks of BMSs are physical disks that have a fixed capacity. If you need to store a large amount of data, you can use SFS, which provides scalable storage.

- GP resources

Table 6-1 Capacities of the cache directories for GP resources

GP Specifications	cache Directory Capacity
GP Vnt1	800 GB
8*GP Vnt1	3 TB
GP Pnt1	800 GB

- CPU resources

  Table 6-2 Capacities of the cache directories for CPU resources

CPU Specifications	cache Directory Capacity
2 vCPUs | 8 GiB	50 GB
8 vCPUs | 32 GiB	50 GB

6.15 Why Do Training Jobs Have Two Hyperparameter Directories /work and /ma-user in ModelArts?

Symptom

The hyperparameter directory for the input and output parameters varies between **/work** and **/ma-user** when creating a training job.

Figure 6-1 /ma-user directory

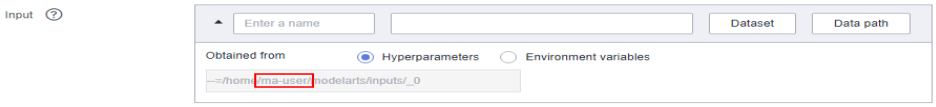
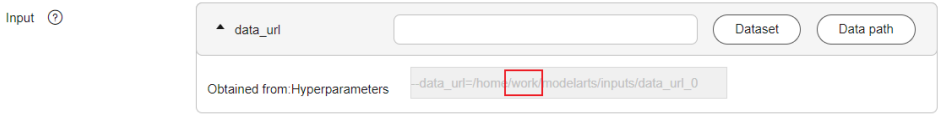


Figure 6-2 /work directory

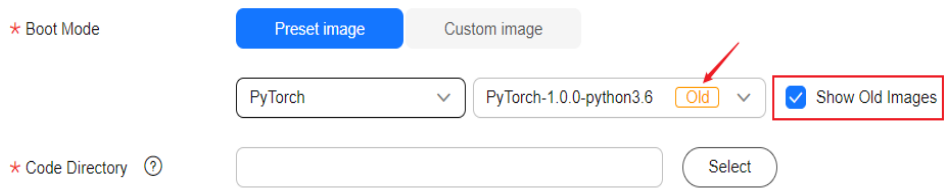


Solution

The directory varies depending on the selected algorithm for the training job.

- If the selected algorithm is created using an old-version image, the hyperparameter directory of the input and output parameters is **/work**.

  Figure 6-3 Creating an algorithm



- If the selected algorithm is not created using an old-version image, the hyperparameter directory of the input and output parameters is **/ma-user**.

6.16 How Do I View the Resource Usage of a Training Job in ModelArts?

On the ModelArts console, choose **Model Training** > **Training Jobs** in the navigation pane on the left to go to the **Training Jobs** page. In the training job list, click a job name to view job details. You can view the following metrics on the **Resource Usages** tab page.

- **CPU**: CPU usage (cpuUsage) percentage (Percent)
- **MEM**: Physical memory usage (memUsage) percentage (Percent)
- **GP**: GP usage (gpUtil) percentage (Percent)
- **GP_MEM**: GP memory usage (gpMemUsage) percentage (Percent)

6.17 How Do I Download a Well Trained Model in ModelArts or Migrate It to Another Account?

You can download a model trained by a training job and upload the downloaded model to OBS in the region corresponding to the target account.

Obtaining a Model Download Path

1. Log in to the ModelArts console. In the navigation pane on the left, choose **Model Training** > **Training Jobs** to go to the **Training Jobs** page.
2. In the training job list, click a job name to view job details.
3. Obtain the **Output Path** on the left, which is the download path of the trained model.

Migrating a Model to Another Account

Use either of the following methods to migrate a trained model to another account:

- Download the trained model and then upload it to the OBS bucket in the region corresponding to the target account.
- Configure a policy for the folder or bucket where the model is stored to authorize other accounts to perform read and write operations. For details, see **Creating a Custom Bucket Policy (Visual Editor)**.

6.18 What Should I Do If RuntimeError: Socket Timeout Is Displayed During Distributed Process Group Initialization using torchrun?

If the **RuntimeError: Socket Timeout** error occurs during the distributed process group initialization using **torchrun**, you can add the following environment variables to create a training job again to view initialization details and further locate the fault.

- LOGLEVEL=INFO
- TORCH_CPP_LOG_LEVEL=INFO
- TORCH_DISTRIBUTED_DEBUG=DETAIL

The **RuntimeError: Socket Timeout** error is caused by a significant time discrepancy between tasks when running the **torchrun** command. The time discrepancy is caused by initialization tasks, like downloading the training data and checkpoint read/write, which happen before the **torchrun** command is run. If the time taken to complete these initialization tasks varies significantly, a Socket Timeout error may occur. When this error happens, check the time difference between the **torchrun** execution points for each task. If the time difference is too large, optimize the initialization process before running the **torchrun** command to ensure a reasonable time gap.

6.19 What Should I Do If an Error Is Reported Indicating that the .so File in the $ANACONDA_DIR/envs/$DEFAULT_CONDA_ENV_NAME/lib Directory Cannot Be Found During Training?

If there is an error indicating that the .so file is unavailable in the **$ANACONDA_DIR/envs/$DEFAULT_CONDA_ENV_NAME/lib** directory, add the directory to **LD_LIBRARY_PATH** and place the following command before the preceding boot command:

```
export LD_LIBRARY_PATH=$ANACONDA_DIR/envs/$DEFAULT_CONDA_ENV_NAME/lib:$LD_LIBRARY_PATH;
```

For example, the example boot command used in method 1 is as follows:

```
export LD_LIBRARY_PATH=$ANACONDA_DIR/envs/$DEFAULT_CONDA_ENV_NAME/lib:$LD_LIBRARY_PATH;
python /home/ma-user/modelarts/user-job-dir/code/train.py
```


7 ModelArts Standard Inference Deployment

7.1 How Do I Import a Keras .h5 Model to ModelArts?

ModelArts does not support the import of models in .h5 format. You can convert the models in .h5 format of Keras to the TensorFlow format and then import the models to ModelArts.

For details about how to convert the Keras format to the TensorFlow format, see the **Keras official website**.

7.2 How Do I Edit the Installation Package Dependency Parameters in the Model Configuration File When Importing a Model to ModelArts?

Symptom

When importing a model from OBS or a container image, edit a model configuration file. The model configuration file describes the model usage, computing framework, precision, inference code dependency package, and model API. The configuration file must be in JSON format. **dependencies** in the model configuration file specifies the dependencies required for configuring the model inference code. This parameter requires the package name, installation method, and version constraints. For details, see **Specifications for Editing a Model Configuration File**. The following section describes how to edit **dependencies** in the model configuration file during model import.

Solution

The installation packages must be installed in sequence. For example, before installing **mmcv-full**, install **Cython**, **pytest-runner**, and **pytest**. In the configuration file, **Cython**, **pytest-runner**, and **pytest** are ahead of **mmcv-full**.

Example:


```
"dependencies": [
  {
  "installer": "pip",
  "packages": [
    {
      "package_name": "Cython"
    },
    {
      "package_name": "pytest-runner"
    },
    {
      "package_name": "pytest"
    },
    {
      "restraint": "ATLEAST",
      "package_version": "5.0.0",
      "package_name": "Pillow"
    },
    {
      "restraint": "ATLEAST",
      "package_version": "1.4.0",
      "package_name": "torch"
    },
    {
      "restraint": "ATLEAST",
      "package_version": "1.19.1",
      "package_name": "numpy"
    },
    {
      "package_name": "mmcv-full"
    }
   ]
  }
]
```

If installing **mmcv-full** failed, the possible cause is that GCC was not installed in the base image, leading to a compilation failure. In this case, use the wheel package on premises to install **mmcv-full**.

Example:

```
"dependencies": [
  {
  "installer": "pip",
  "packages": [
    {
      "package_name": "Cython"
    },
    {
      "package_name": "pytest-runner"
    },
    {
      "package_name": "pytest"
    },
    {
      "restraint": "ATLEAST",
      "package_version": "5.0.0",
      "package_name": "Pillow"
    },
    {
      "restraint": "ATLEAST",
      "package_version": "1.4.0",
      "package_name": "torch"
    },
    {
      "restraint": "ATLEAST",
      "package_version": "1.19.1",
      "package_name": "numpy"
    },
```



```
    {
      "package_name": "mmcv_full-1.3.9-cp37-cp37m-manylinux1_x86_64.whl"
    }
   ]
  }
]
```

dependencies in the model configuration file supports multiple dependency structure arrays in list format.

Example:

```
"dependencies": [
  {
  "installer": "pip",
  "packages": [
    {
      "package_name": "Cython"
    },
    {
      "package_name": "pytest-runner"
    },
    {
      "package_name": "pytest"
    },
    {
      "package_name": "mmcv_full-1.3.9-cp37-cp37m-manylinux1_x86_64.whl"
    }
   ]
  },
  {
  "installer": "pip",
  "packages": [
    {
      "restraint": "ATLEAST",
      "package_version": "5.0.0",
      "package_name": "Pillow"
    },
    {
      "restraint": "ATLEAST",
      "package_version": "1.4.0",
      "package_name": "torch"
    },
    {
      "restraint": "ATLEAST",
      "package_version": "1.19.1",
      "package_name": "numpy"
    },
   ]
  }
]
```

7.3 How Do I Change the Default Port When I Create a Real-Time Service Using a Custom Image in ModelArts?

A port number (for example, 8443) has been specified in a model configuration file. If you do not specify a port (default port 8080 will be used then) or specify another port during AI application creation, deploying the AI application as a service will fail. In this case, set the port number to 8443 in the AI application to resolve this issue.

To change the default port, do as follows:

1. Log in to the ModelArts management console. In the navigation pane, choose **AI Application Management** > **AI Applications**.
2. Click **Create**. On the page for creating an AI application, set **Meta Model Source** to **Container image** and select a custom image.
3. Configure the container API and port number. Ensure that the port number is the same as that specified in the model configuration file.
4. After the configuration, click **Create now**. Wait until the AI application runs properly.
5. Deploy the AI application as a real-time service again.

7.4 Does ModelArts Support Multi-Model Import?

Importing a model package from OBS to ModelArts applies to single-model scenarios.

If multiple models are required, you are advised to import custom images from SWR to create models and deploy services.

For details about how to create a custom image, see **Creating a Custom Image and Using It to Create an AI Application**.

7.5 What Are the Restrictions on the Image Size for Importing AI Applications to ModelArts?

ModelArts uses containers for deploying services. There are size limitations during container runtime. If the size of your model file, custom file, or system file exceeds the container engine space, a message will be displayed, indicating that the image space is insufficient.

The maximum container engine space in a public resource pool is 50 GB, and that for a dedicated resource pool is 50 GB by default. You can set the container engine space for a dedicated resource pool when you create it, which does not increase costs.

If the AI application is imported from OBS or a training job, the total size of the base image, model files, code, data files, and software packages cannot exceed the limit.

If the AI application is imported from a custom image, the total size of the decompressed image and image dependencies cannot exceed the limit.

7.6 What Are the Differences Between Real-Time Services and Batch Services in ModelArts?

- Real-Time Services

  Models are deployed as web services. You can access the services through the management console or APIs.
- Batch Services

A batch service performs inference on batch data and automatically stops after data processing is completed.

A batch service processes batch data at a time. A real-time service provides APIs for you to call the service for inference.

7.7 Why Can't I Select Ascend Snt3 Resources When Deploying Models in ModelArts?

Ascend Snt3 resources are limited. If resources are sold out, you cannot select Ascend Snt3 resources (in the public resource pool) for inference during deployment. On the **Deploy** page, the **Ascend: 1*Ascend-Snt3 (8 GB) | ARM: 3 vCPUs, 6 GB** resource will be unavailable.

Solutions:

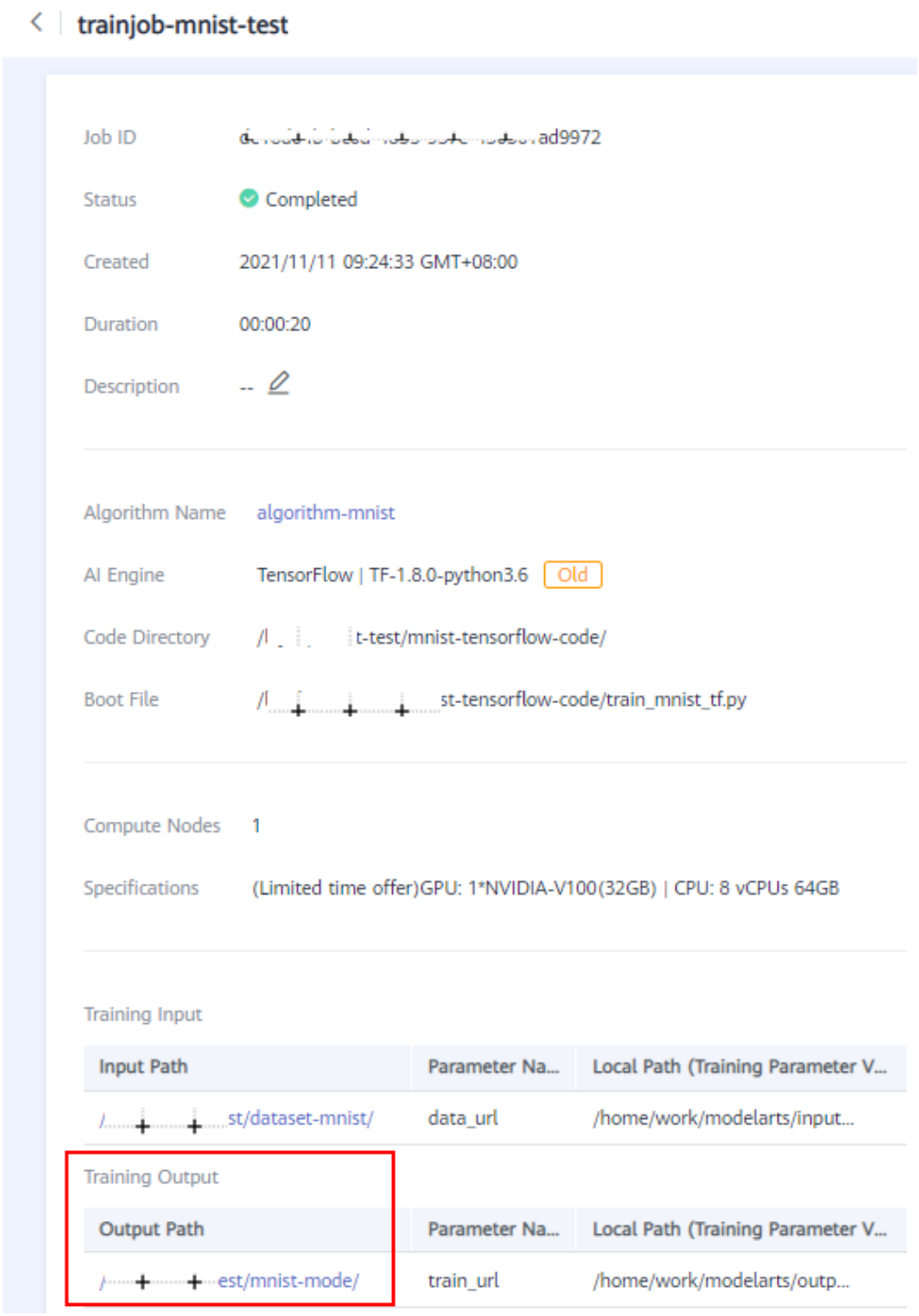
- Method 1: If you want to use Ascend Snt3 in the public resource pool, you can wait for other users to release the resources. If other services using the Ascend Snt3 resources stop, you can select the resources for deployment.
- Method 2: If you have a dedicated resource pool with Ascend Snt3 resources, you can create an Ascend Snt3 dedicated resource pool.
- Method 3: If Ascend Snt3 resources in the dedicated resource pool are sold out, you can create an Ascend Snt3 dedicated resource pool after other users have deleted their Ascend Snt3 instances.

7.8 Can I Locally Deploy Models Trained on ModelArts?

Models trained using ModelArts built-in algorithms are stored in OBS buckets and can be downloaded to a local directory.

1. In the training job list, click the name of the target training job to go to its details page, on which you can obtain the training output path.

Figure 7-1 Training output path



2. Click the path to go to the OBS object path. Then, download the model from OBS.
3. Deploy the downloaded model locally.

   For details, see **Creating a Local Model** and **Debugging a Service**.

7.9 What Is the Maximum Size of a ModelArts Real-Time Service Prediction Request Body?

After a service is deployed and running, you can send an inference request to the service. The requested content can be text, images, audios, or videos, depending on the model of the service.

If you perform prediction by calling an inference request address (URL of Huawei Cloud APIG) displayed on the **Usage Guides** tab of the service details page, the maximum size of the request body is 12 MB. If the request body is oversized, the request will be intercepted.

If you perform the prediction on the **Prediction** tab of the service details page, the size of the request body cannot exceed 8 MB. The size limit varies between the two tab pages because they use different network links.

Ensure that the size of a request body does not exceed the upper limit. If there are high-concurrency and heavy-traffic inference requests, submit a service ticket to professional service support.

7.10 How Do I Prevent Python Dependency Package Conflicts in a Custom Prediction Script When Deploying a Real-Time Service in ModelArts?

Before importing a model, save the inference code and configuration file in the model folder. When coding with Python, import custom packages in relative import (Python import) mode.

If there are packages with duplicate names in the ModelArts inference framework code and they are imported not in relative import mode, a conflict will occur, leading to a service deployment or prediction failure.

7.11 How Do I Speed Up Real-Time Service Prediction in ModelArts?

- When deploying a real-time service, select instance specifications with better performance for faster prediction. For example, use GPs instead of CPUs.
- When deploying a real-time service, add the number of instances.

  If you set the number of instances to **1**, the standalone computing mode is used. If you set the number of instances to a value greater than **1**, the distributed computing mode is used. Configure this parameter based on site requirements.
- The inference speed is closely related to the model complexity. Try to optimize the model for faster prediction.

  ModelArts provides model version management to facilitate source tracing and repeated model tuning.

7.12 Can a New-Version AI Application Still Use the Original API in ModelArts?

ModelArts supports multiple model versions and flexible traffic policies to smoothly gray upgrade model versions. After a service is modified to deploy a new-version model or its model version is upgraded, the original service prediction API remains unchanged.

To adjust a model version, perform the operations described in this section.

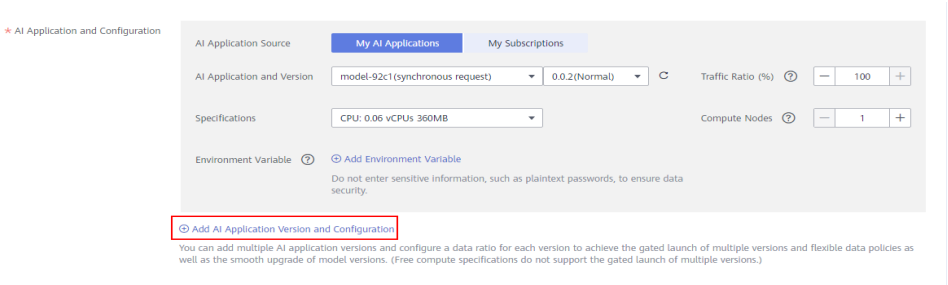
Prerequisites

- A service has been deployed.
- A **new-version AI application** has been created.

Procedure

1. Log in to the ModelArts management console. In the navigation pane on the left, choose **Service Deployment** > **Real-Time Services**. The **Real-Time Services** page is displayed.
2. Locate the target service and click **Modify** in the **Operation** column. The **Modify Service** page is displayed.
3. In the **AI Application and Configuration** area, click **Add AI Application Version and Configuration** to add a new version.

   Figure 7-2 Add AI Application Version and Configuration

4. Set the traffic proportion of the two versions. Service calling requests are allocated based on the proportion. For details about other settings, see **Parameters**. After the setting, click **Next**.
5. Confirm the information and click **Submit**.

7.13 What Is the Format of a Real-Time Service API in ModelArts?

After an AI application is deployed as a real-time service, you can use the API for inference.

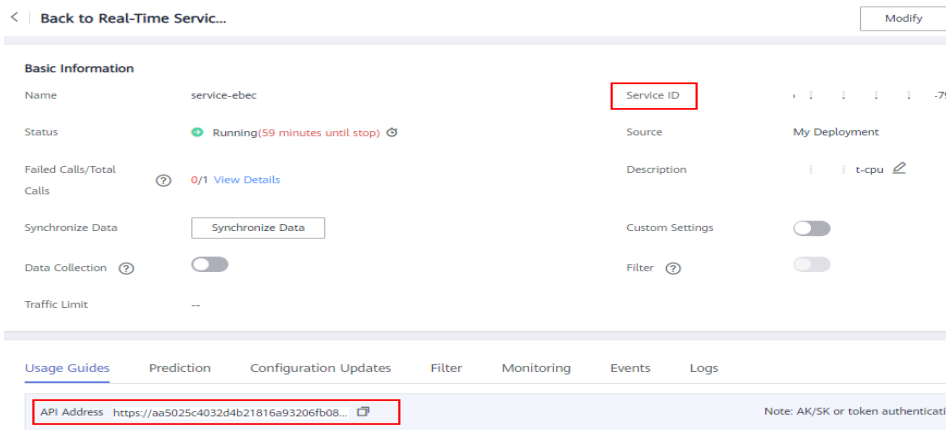
The format of an API is as follows:

```
https://Domain name/Version/infer/service ID
```


Example:

```
https://6ac81cdfac4f4a30be95xxxbb682.apig.xxx.xxx.com/v1/infers/
468d146d-278a-4ca2-8830-0b6fb37d3b72
```

Figure 7-3 API



7.14 How Do I Fill in the Request Header and Request Body When a ModelArts Real-Time Service Is Running?

Symptom

After a real-time service is deployed, you can obtain its inference request address on the **Usage Guides** tab of the service details page when the service is running. However, there is no instruction for filling in the header and body of an inference request.

Possible Causes

The inference request address on the **Usage Guides** tab of the service details page can be called for inference. For security purposes, ModelArts takes authentication and authorization measures to prevent unauthorized calling of the real-time service. Therefore, the header of a prediction request contains the identity information of the request initiator, and the body contains the content to be predicted.

The header must be authenticated by following HUAWEI CLOUD authentication rules. The body must be configured based on model requirements, such as the requirements of pre-processing scripts or custom images.

Solution

- Header:

  On the **Usage Guides** tab of the service details page, you can obtain a maximum of two API addresses, one for IAM or AK/SK authentication and the other for application authentication. The header structure varies depending on the authentication mode.

  - IAM or AK/SK authentication: In the header, enter the domain-level token of the tenant in the target region in the **X-Auth-Token** field. For details, see **Obtaining a User Token Through Password Authentication**.
  - Application authentication: Application authentication can be further classified as AppCode authentication and application signature authentication.
    - For AppCode authentication, enter the AppCode of the application associated with the real-time service in the **X-Apig-AppCode** field of the header.
    - For application signature authentication, in the header, enter the **X-Sdk-Date** and **Authorization** values generated using the AppKey and AppSecret of the application associated with the real-time service through the SDK or tool to authenticate the signature of the request. For details, see **Access Authenticated Using an Application**.
- Body:

  The body varies depending on the model source.
  - If the model is imported from a container image, the body must be configured based on the custom image requirements. For details, contact the image creator.
  - If the model is imported from OBS, the requirements on the body are reflected in inference code preprocessing, which will convert the input HTTP body into the input required by the model. For details, see **Specifications for Model Inference Coding**.
  - If the model is obtained from AI Gallery, check the calling description in AI Gallery or consult the model provider.

Summary and Suggestions

None

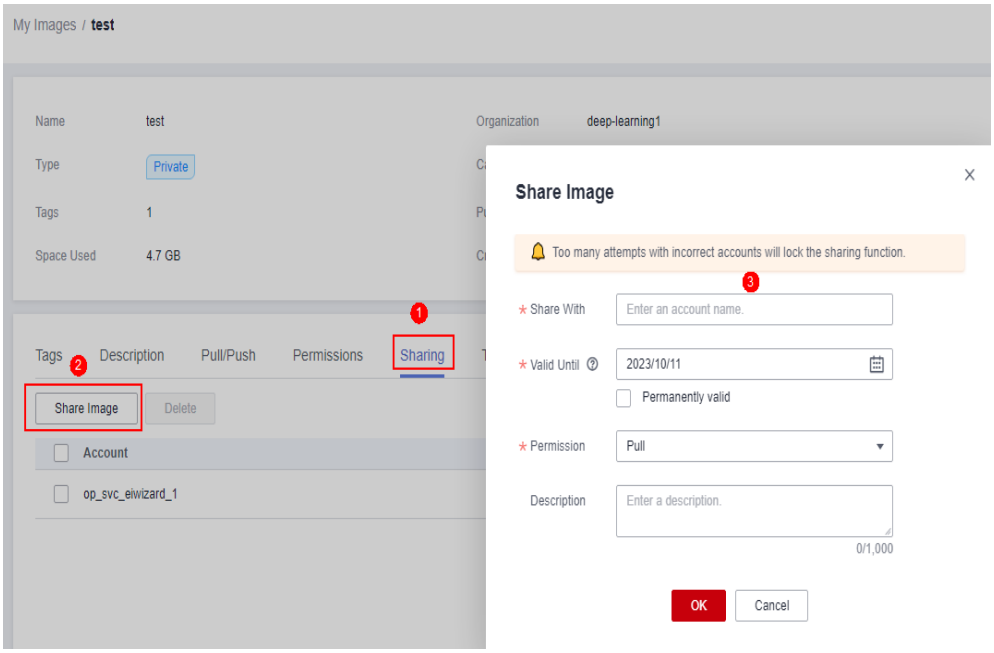
8 ModelArts Standard Images

8.1 How Do I Use the Image Customized by a User Under a Different Tenant Account to Create a Notebook Instance?

Two users belong to different tenant accounts. If user A needs to use user B's custom image to create a notebook instance, user B needs to share the image with user A. Then user A pulls the shared image and registers it before using it in the notebook instance. The procedure is as follows:

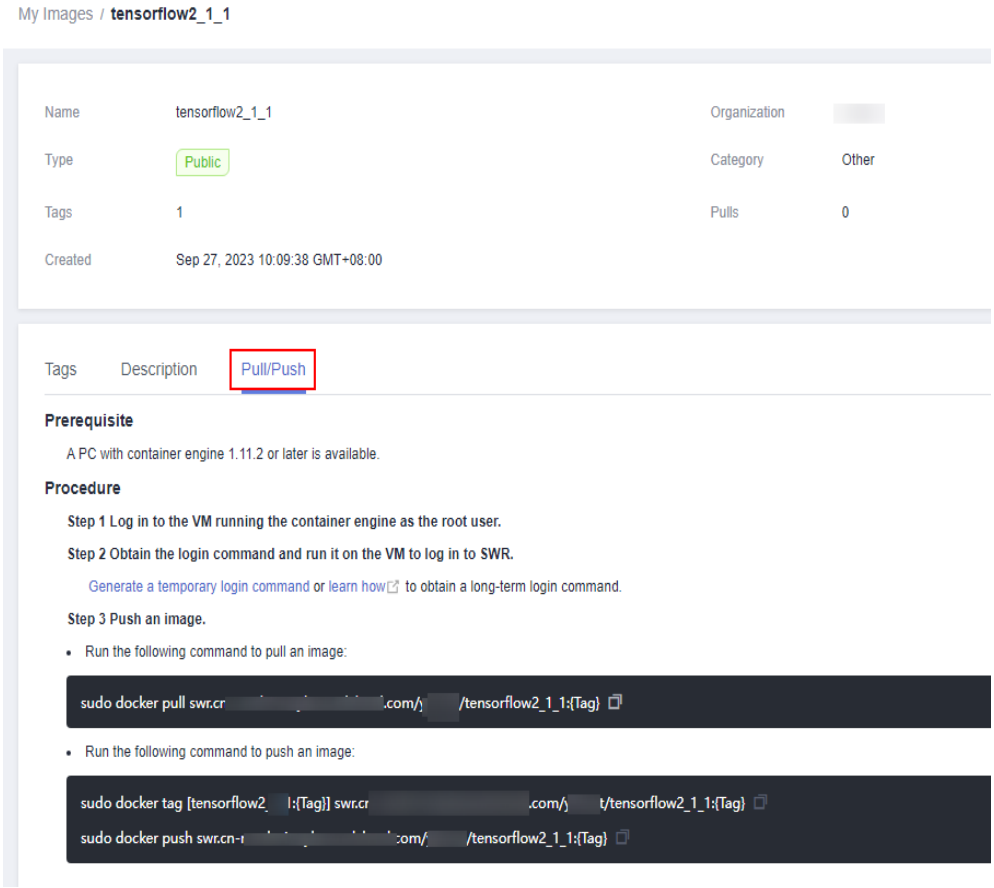
User B's operations:

1. Log in to the SWR console and choose **My Images**.
2. Click the name of the image to be shared to access its details page.
3. On the **Sharing** tab, click **Share Image**. In the displayed dialog box, set required parameters such as the account and click **OK**.

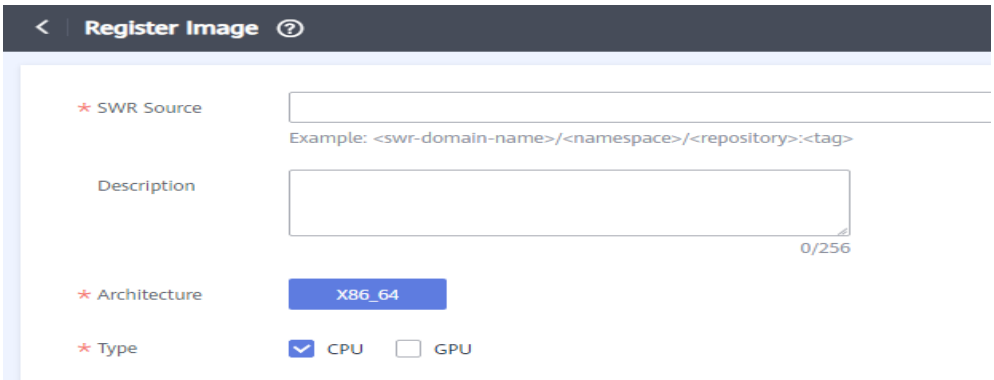


User A's operations:

1. Log in to the SWR console, choose **My Images** > **Shared Images**, view the image shared by user B, and click the image name to go to its details page.
2. Pull the image shared by user B as your own image by following the instructions provided on the **Pull/Push** tab.



3. Log in to the ModelArts console, select the pulled image, and register it. After the registration is successful, you can use the image on the notebook instance.



8.2 How Do I Log In to SWR and Upload Images to It?

This section describes how to log in to SWR and upload images to it.

Step 1 Log In to SWR

1. Log in to the SWR console and select the target region.
2. Click **Create Organization** in the upper right corner and enter an organization name to create an organization. **deep-learning** is used as an example. Replace it in subsequent commands with the actual organization name.
3. Click **Generate Login Command** in the upper right corner to obtain a login command.
4. Log in to the ECS as user **root** and enter the login command.

Figure 8-1 Login command executed on the ECS

Step 2 Upload Images to SWR

Do as follows to upload an image to SWR:

1. Log in to SWR and tag the image to be uploaded. Replace the organization name **deep-learning** in the following command with the actual organization name obtained in step 1.
   ```
   sudo docker tag tf-1.13.2:latest swr.example.com/deep-learning/tf-1.13.2:latest
   ```
2. Run the following command to upload the image:
   ```
   sudo docker push swr.example.com/deep-learning/tf-1.13.2:latest
   ```

Figure 8-2 Uploading an Image

3. After the image is uploaded, choose **My Images** in navigation pane on the left of the SWR console to view the uploaded image.

   swr.example.com/deep-learning/tf-1.13.2:latest is the SWR URL of the custom image.

8.3 How Do I Configure Environment Variables for an Image in a Dockerfile?

In a Dockerfile, use the ENV instruction to configure environment variables. For details, see **Dockerfile reference**.

8.4 How Do I Start a Container Using a Docker Image?

An image saved using a notebook instance contains the **Entrypoint** parameter, as shown in **Figure 8-3**. The executable file or command specified in the **Entrypoint** parameter will overwrite the default boot command of the image. The command input in the **Entrypoint** parameter is not preset in the image. When you run **docker run** in the local environment to start the image, an error message is displayed, indicating that the container creation task fails because the boot file or directory is not found, as shown in **Figure 8-4**.

To avoid this error, configure the **--entrypoint** parameter to overwrite the program specified in **Entrypoint**. Use the boot file or command specified by the **--entrypoint** parameter to start the image. Example:

```
docker run -it -d --entrypoint /bin/bash image:tag
```

Figure 8-3 Entrypoint parameter

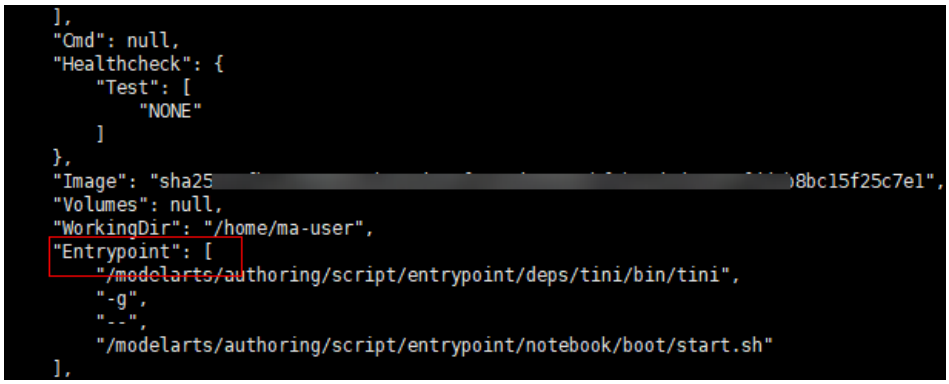
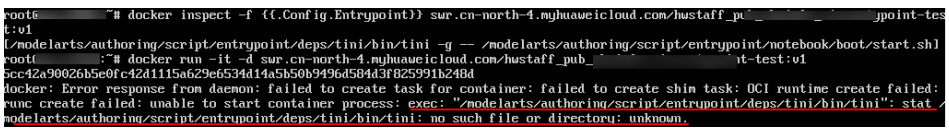


Figure 8-4 Error reported when an image is being started



8.5 How Do I Configure a Conda Source on a ModelArts Notebook Instance?

You can install the development dependencies on Notebook instances as you need. Package management tools pip and Conda can be used to install regular dependencies. The pip source has been configured and can be used directly for installation, while the Conda source requires further configuration.

This section describes how to configure the Conda source on a notebook instance.

Configuring the Conda Source

The Conda software has been preset in images.

Common Conda Commands

For details about all Conda commands, see **Conda official documents**. The following table lists only common commands.

Table 8-1 Common Conda commands

Description	Command
Obtain online help.	conda --help conda update --help # Obtain help for a command, for example, update.
View the Conda version.	conda -V
Update Conda.	conda update conda # Update Conda. conda update anaconda # Update Anaconda.
Manage environments.	conda env list # Show all virtual environments. conda info -e # Show all virtual environments. conda create -n myenv python=3.7 # Create an environment named myenv with Python version 3.7. conda activate myenv # Activate the myenv environment. conda deactivate # Disable the current environment. conda remove -n myenv --all # Delete the myenv environment. conda create -n newname --clone oldname # Clone the old environment to the new environment.
Manage packages.	conda list # Check the packages that have been installed in the current environment. conda list -n myenv # Specify the packages installed in the myenv environment. conda search numpy # Obtain all information of the numpy package. conda search numpy=1.12.0 --info # View the information of NumPy 1.12.0. conda install numpy pandas # Concurrently install the NumPy and Pandas packages. conda install numpy=1.12.0 # Install NumPy of a specified version. # The install, update, and remove commands use -n to specify an environment, and the install and update commands use -c to specify a source address. conda install -n myenv numpy # Install the numpy package in the myenv environment. conda install -c https://conda.anaconda.org/anaconda numpy # Install NumPy using https://conda.anaconda.org/anaconda. conda update numpy pandas # Concurrently update the NumPy and Pandas packages. conda remove numpy pandas # Concurrently uninstall the NumPy and Pandas packages. conda update –-all # Update all packages in the current environment.
Clear Conda.	conda clean -p # Delete useless packages. conda clean -t # Delete compressed packages. conda clean -y --all # Delete all installation packages and clear caches.

Saving a Notebook Instance as an Image

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

packages are installed on a notebook instance, it is a good practice to save the instance as an image to prevent the dependency packages from being lost.

8.6 What Are the Software Version Requirements for a Custom Image?

When customizing an image, ensure that the software libraries you use (such as NCCL, CUDA, and OFED) match those available in ModelArts. The software version in your image must meet the following requirements:

- NCCL 2.7.8 or later
- OFED MLNX_OFED_LINUX-5.4-3.1.0.0 or later
- The CUDA version needs to be adapted with the GP driver version of the dedicated resource pool. To obtain the GP driver version, go to the dedicated resource pool details page.

8.7 Why Is an Image Reported as Larger Than 35 GB When I'm Saving It But Its Size Is Displayed as 13 GB in SWR?

Symptom

I use an image that is displayed as only about 13 GB in SWR to create a notebook instance in ModelArts. After I start the notebook instance, install some packages, and try to save it as an image, an error occurs indicating that the image is larger than 35 GB and fails to be saved.

Possible Causes

The image size displayed in SWR is the size of the compressed image. The actual size after decompression is 2.5 to 3 times that of the displayed size. Therefore, the image size exceeds 35 GB after a few packages are installed.

8.8 How Do I Prevent the Save Failure of a Custom Image Larger Than 35 GB?

Note the following issues:

1. Select a small base image to create a notebook instance. In this case, there is enough space to install packages. The base image size displayed on SWR should not be larger than 6 GB.
2. Save the image in a directory other than **/home/ma-user/work** under **/home/ma-user**. Save the datasets in the **work** directory.
3. Avoid saving instances as images too frequently. Instead, install all necessary packages at a time and then save the image. Each time you save the image, it increases in size, and Docker's storage mechanism prevents resizing by clearing the disk.

8.9 How Do I Reduce the Size of the Target Image Created on the Local PC or ECS?

Choose a smaller image that fits your needs. For instance, if you are creating a PyTorch 2.1+CUDA 12.2 image and cannot find an exact match from the official website, preferentially select an image that does not have either the PyTorch environment or CUDA installed instead of selecting an image with neither installed (like MindSpore+CUDA 11.*X*). Otherwise, both the base and target images will end up large.

You can take the following measures to reduce the size of an image:

- Reduce layers of the target image.

  If two pip packages **six** and **numpy** need to be installed, install them at the same layer.

  Recommended:

  ```
  RUN pip install six &&\
      pip install numpy
  ```

  Not recommended:

  ```
  RUN pip install six
  RUN pip install numpy
  ```

  An image with more layers takes up more space.

- Install and uninstall the packages at the same layer.

  For example, to uninstall an SCC package downloaded from the official website, do as follows:

  Recommended:

  ```
  RUN mkdir -p /tmp/scc && \
      cd /tmp/scc && \
      wget http://100.95.151.167:6868/aarch64/euler/dls-release/euleros-arm/compiled-software/
  seccomponent-1.1.0-release.aarch64.rpm && \
      rpm -ivh /tmp/scc/seccomponent-1.1.0-release.aarch64.rpm --force --nodeps && \
      rm -rf /tmp/scc
  ```

  Not recommended:

  ```
  RUN mkdir -p /tmp/scc && \
      cd /tmp/scc && \
      wget http://100.95.151.167:6868/aarch64/euler/dls-release/euleros-arm/compiled-software/
  seccomponent-1.1.0-release.aarch64.rpm && \
      rpm -ivh /tmp/scc/seccomponent-1.1.0-release.aarch64.rpm --force --nodeps
  RUN rm -rf /tmp/scc
  ```

8.10 Will an Oversized Image Become Smaller If I Uninstall and Reinstall Its Packages?

No, it will get larger. The new image is derived from the original one without any change in size, as the number of layers remains the same. Removing packages or datasets will only add more layers, resulting in an increase in size.

8.11 What Do I Do If Error "ModelArts.6787" Is Reported When I Register an Image in ModelArts?

Symptom

When a user creates an image on the image management page, an error is displayed indicating that **ModelArts.6787: Image** ***** cannot be used, the image cannot be found in ***** on SWR, and the user should check the image and access permission configuration on the SWR console or try with another image.

Possible Causes

The possible causes are as follows:

- The image is a private image registered by the tenant account and is used by an IAM account. However, the tenant account did not assign SWR permissions to the IAM account. As a result, the IAM account cannot find the image on the SWR console.
- The image does not belong to either the tenant account or the IAM account. It is a public image shared by others and has been deleted by the owner.
- The image does not belong to either the tenant account or the IAM account. It is a public image shared by others and is set to private by the owner.

Solution

Rectify the fault based on the cause. For cause 1, use the tenant account to grant the SWR permissions to the IAM account. For cause 2, contact the SWR image owner to check whether the image exists. For cause 3, contact the SWR image owner to confirm the image attribute.

8.12 How Do I Set the Default Kernel?

If you want to open a notebook instance with your custom kernel as the default one, do as follows:

Solution:

1. Run the following command on the Terminal to specify environment variables in the image:
   ```
   # python-3.7.10 indicates the kernel to be set.
   export KG_DEFAULT_KERNEL_NAME=python-3.7.10
   ```
2. Click **More** in the **Operation** column and select **Save Image**. After the image is saved, restart the notebook instance.

9 ModelArts Standard Dedicated Resource Pools

9.1 Can I Use ECSs to Create a Dedicated Resource Pool for ModelArts?

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

9.2 Can I Deploy Multiple Services on One Dedicated Resource Pool Node in ModelArts?

Yes. This operation is allowed.

When you deploy services, select a dedicated resource pool and customize the compute node flavor. Select the node with a low flavor. When the resource pool node allows multiple service node flavors, multiple services can be deployed. If you use this method to deploy a model for inference, ensure that the selected flavor complies with the minimum model requirements for inference. Otherwise, the deployment or prediction may fail.

9.3 What Are the Differences Between Public Resource Pools and Dedicated Resource Pools in ModelArts?

- **Dedicated resource pool**: It delivers more controllable resources and cannot be shared with other users. Create a dedicated resource pool and select it during AI development.
- **Public resource pool**: It provides large-scale public computing clusters, which are allocated based on job parameter settings. Resources are isolated by job. You can use ModelArts public resource pools to deliver training jobs, deploy models, or run DevEnviron instances and will be billed on a pay-per-use basis.

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

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

9.4 Why Does a Job in ModelArts Stay in the Pending State?

First in first out (FIFO) applies to training jobs. Subsequent jobs can be executed only after the preceding job is complete. This may lead to starvation of small jobs.

NOTE

Job starvation is as follows: For example, a 64-card training job is queuing, and a 1-card training job follows the 64-card one. The 1-card training job can be executed only after the resources of 64 cards are idle. Even if the resources of 30 cards are available, the 1-card training job cannot be executed.

- If the resource pool is a public one, the job is generally pending because resources are occupied other users. Try to take the following measures:
  - If a free flavor was used, change it to a charged one. Few resources are provided for free flavors, leading to a high queuing probability.
  - To minimize queuing, select flavors with the smallest number of cards possible. This reduces the likelihood of queuing significantly. For example, the probability of queuing when selecting a 1-card flavor is much less than that of queuing when selecting an 8-card flavor.
  - Switch to another region.
  - If resources will be used for a long term, purchase a dedicated resource pool.
- If a dedicated resource pool is used, perform the following operations:
  a. Check whether other jobs (including inference jobs, training jobs, and development environment jobs) are running in the dedicated resource pool.

     On the **Overview** page, you can go to the details page of the running jobs or instances to check whether the dedicated resource pool is used. You can stop them based on your needs to release resources.

b. Click the dedicated resource pool to go to the details page and view the job list.

   If other jobs are waiting in the queue, the new job must also join the queue.

c. Check whether resources are fragmented.

   For example, the cluster has two nodes, and there are four idle cards on each node. However, your job requires eight cards on one node. In this case, the idle resources cannot be allocated to your job.

9.5 Why Can I View the Deleted Dedicated Resource Pools That Failed to Be Created on the ModelArts Console?

After a dedicated resource pool is deleted on the console, the backend releases the resources used by the pool. It takes several minutes to release the resources, during which the pool is still displayed on the console. To create a dedicated resource pool again, wait 5 minutes after the deletion. Additionally, do not use the name of the dedicated resource pool that fails to be created to name the new dedicated resource pool. To perform an automated test on the UI, it is a good practice to use a random string as the name of the created dedicated resource pool.

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

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

For details about how to interconnect with a VPC, see **Interconnecting a VPC with a ModelArts Network**.

10 ModelArts Studio (MaaS)

10.1 How Long Does It Take for an API Key to Become Valid After It Is Created in MaaS?

A MaaS API key becomes valid a few minutes after creation.

10.2 Can I Use a MaaS API Key Across Regions?

API keys work only in their specific region. For example, an API key created in the CN South-Hong Kong region can only be used in this region. The same rule applies to API keys created in other regions.

10.3 What Are the Format Requirements for Configuring the Model Service API URL in MaaS?

When setting up the model service API URL on most platforms, remove the **/chat/completion**s part at the end.

For example, change **https://example.com/v1/chat/completions** to **https://example.com/v1**.

To obtain the model service API URL, follow these steps:

1. Log in to the ModelArts Studio console and choose **Real-Time Inference** in the navigation pane on the left.
2. In the **Real-Time Inference** > **My Services** tab, choose **More** > **View Call Description** in the **Operation** column of the target running service.
3. On the displayed dialog box, obtain the model service API URL required for calling the service.

10.4 How Do I Obtain the Model Name in MaaS?

Use the model name displayed on the MaaS console's **View Call Description** page, not the service name. The model name might differ from the service name.

Copying the service name can cause the call to fail. Check the information carefully.

To obtain the model name, follow these steps:

1. Log in to the ModelArts Studio console and choose **Real-Time Inference** in the navigation pane on the left.
2. In the **Real-Time Inference** > **My Services** tab, choose **More** > **View Call Description** in the **Operation** column of the target running service.
3. On the displayed dialog box, obtain the model name required for calling the service.

11 API/SDK

11.1 Can ModelArts APIs or SDKs Be Used to Download Models to a Local PC?

ModelArts APIs or SDKs cannot be used to download models to a local PC. However, the output models of training jobs are stored in OBS. You can use OBS APIs or SDKs to download the models. For details, see **Downloading an Object**.

11.2 Does ModelArts Use the OBS API to Access OBS Files over an Intranet or the Internet?

In the same region, ModelArts uses the OBS API to access files stored in OBS over an intranet and does not consume public network traffic.

If you download data from OBS through the Internet, you will be charged for the OBS public network traffic. For details about OBS billing, see **Billing Items**.

12 History

12.1 How Do I Upload Data to OBS?

Before using ModelArts to develop AI models, data needs to be uploaded to an OBS bucket. You can log in to the OBS console to create an OBS bucket, create a folder in it, and upload data. For details about how to upload data, see ***Object Storage Service Getting Started***.

12.2 Which AI Frameworks Does ModelArts Support?

The AI frameworks and versions supported by ModelArts vary slightly based on the development environment notebook, training jobs, and model inference (AI application management and deployment). The following describes the AI frameworks supported by each module.

Development Environment Notebook

The image and versions supported by development environment notebook instances vary based on runtime environments.

Table 12-1 Images supported by notebook

Image	Description	Supported Chip	Remote SSH	Online Jupyter Lab
pytorch1.8-cuda10.2-cudnn7-ubuntu18.04	CPU- or GPU-powered public image for general algorithm development and training, with built-in AI engine PyTorch 1.8	CPU or GPU	Yes	Yes

Image	Description	Supported Chip	Remote SSH	Online Jupyter Lab
mindspore1.7.0-cuda10.1-py3.7-ubuntu18.04	CPU- or GPU-powered general algorithm development and training, preconfigured with AI engine MindSpore 1.7.0 and CUDA 10.1	CPU or GPU	Yes	Yes
mindspore1.7.0-py3.7-ubuntu18.04	CPU-powered general algorithm development and training, preconfigured with AI engine MindSpore 1.7.0	CPU	Yes	Yes
pytorch1.10-cuda10.2-cudnn7-ubuntu18.04	CPU- or GPU-powered general algorithm development and training, preconfigured with AI engine PyTorch 1.10 and CUDA 10.2	CPU or GPU	Yes	Yes
tensorflow2.1-cuda10.1-cudnn7-ubuntu18.04	CPU and GPU general algorithm development and training, preconfigured with AI engine TensorFlow2.1	CPU or GPU	Yes	Yes
conda3-ubuntu18.04	Clean customized base image only includes Conda	CPU	Yes	Yes
pytorch1.4-cuda10.1-cudnn7-ubuntu18.04	CPU- or GPU-powered public image for general algorithm development and training, with built-in AI engine PyTorch 1.4	CPU or GPU	Yes	Yes

Image	Description	Supported Chip	Remote SSH	Online Jupyter Lab
tensorflow1.13-cuda10.0-cudnn7-ubuntu18.04	GPU-powered public image for general algorithm development and training, with built-in AI engine TensorFlow 1.13.1	GPU	Yes	Yes
conda3-cuda10.2-cudnn7-ubuntu18.04	Clean customized base image includes CUDA 10.2, Conda	CPU	Yes	Yes
spark2.4.5-ubuntu18.04	CPU-powered algorithm development and training, preconfigured with PySpark 2.4.5 and can be attached to preconfigured Spark clusters including MRS and DLI	CPU	No	Yes
mindspore1.2.0-cuda10.1-cudnn7-ubuntu18.04	GPU-powered public image for algorithm development and training, with built-in AI engine MindSpore-GPU	GPU	Yes	Yes
mindspore1.2.0-openmpi2.1.1-ubuntu18.04	CPU-powered public image for algorithm development and training, with built-in AI engine MindSpore-CPU	CPU	Yes	Yes
pytorch_1.11.0-cann_7.0.1-py_3.9-euler_2.10.7-aarch64-snt9b	Ascend_snt9b and Arm-powered public image for algorithm development and training, with built-in AI engine PyTorch 1.11	Ascend_snt9b	Yes	Yes

Image	Description	Supported Chip	Remote SSH	Online Jupyter Lab
mindspore_2.2.0-cann_7.0.1-py_3.9-euler_2.10.7-aarch64-snt9b	Ascend_snt9b and Arm-powered public image for algorithm development and training, with built-in AI engine MindSpore	Ascend_snt9b	Yes	Yes
pytorch_2.1.0-cann_7.0.1-py_3.9-euler_2.10.7-aarch64-snt9b	Ascend_snt9b and Arm-powered public image for algorithm development and training, with built-in AI engine PyTorch 2.1	Ascend_snt9b	Yes	Yes

Training Jobs

The following table lists the AI engines.

The built-in training engines are named in the following format:
<Training engine name_version>-[cpu | <cuda_version | cann_version >]-<py_version>-<OS name_version>-<x86_64 | aarch64>

Table 12-2 AI engines supported by training jobs

Runtime Environment	System Architecture	System Version	AI Engine and Version	Supported CUDA or Ascend Version
TensorFlow	x86_64	Ubuntu18.04	tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64	cuda10.1
PyTorch	x86_64	Ubuntu18.04	pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64	cuda10.2
MPI	x86_64	Ubuntu18.04	mindspore_1.3.0-cuda_10.1-py_3.7-ubuntu_1804-x86_64	cuda_10.1

Runtime Environment	System Architecture	System Version	AI Engine and Version	Supported CUDA or Ascend Version
Horovod	x86_64	ubuntu_18.04	horovod_0.20.0-tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64	cuda_10.1
			horovod_0.22.1-pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64	cuda_10.2

NOTE

Supported AI engines vary depending on regions.

Supported AI Engines for ModelArts Inference

If you import a model from a template or OBS to create an AI application, the following AI engines and versions are supported.

NOTE

- Runtime environments marked with **recommended** are unified runtime images, which will be used as mainstream base inference images. The installation packages of unified images are richer. For details, see Base Inference Images.
- Images of the old version will be discontinued. Use unified images.
- The base images to be removed are no longer maintained.
- Naming a unified runtime image: <*AI engine name and version*> - <*Hardware and version*: CPU, CUDA, or CANN> - <*Python version*> - <*OS version*> - <*CPU architecture*>

Table 12-3 Supported AI engines and their runtime

Engine	Runtime	Note
TensorFlow	python3.6 python2.7 (unavailable soon) tf1.13-python3.6-gpu tf1.13-python3.6-cpu tf1.13-python3.7-cpu tf1.13-python3.7-gpu tf2.1-python3.7 (unavailable soon) tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64 (recommended)	• TensorFlow 1.8.0 is used in **python2.7** and **python3.6**. • **python3.6**, **python2.7**, and **tf2.1-python3.7** indicate that the model can run on both CPUs and GPUs. For other runtime values, if the suffix contains **cpu** or **gpu**, the model can run only on CPUs or GPUs. • The default runtime is **python2.7**.
Spark_MLlib	python2.7 (unavailable soon) python3.6 (unavailable soon)	• Spark_MLlib 2.3.2 is used in **python2.7** and **python3.6**. • The default runtime is **python2.7**. • **python2.7** and **python3.6** can only be used to run models on CPUs.
Scikit_Learn	python2.7 (unavailable soon) python3.6 (unavailable soon)	• Scikit_Learn 0.18.1 is used in **python2.7** and **python3.6**. • The default runtime is **python2.7**. • **python2.7** and **python3.6** can only be used to run models on CPUs.
XGBoost	python2.7 (unavailable soon) python3.6 (unavailable soon)	• XGBoost 0.80 is used in **python2.7** and **python3.6**. • The default runtime is **python2.7**. • **python2.7** and **python3.6** can only be used to run models on CPUs.

Engine	Runtime	Note
PyTorch	python2.7 (unavailable soon) python3.6 python3.7 pytorch1.4-python3.7 pytorch1.5-python3.7 (unavailable soon) pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64 (recommended)	• PyTorch 1.0 is used in **python2.7**, **python3.6**, and **python3.7**. • **python2.7**, **python3.6**, **python3.7**, **pytorch1.4-python3.7**, and **pytorch1.5-python3.7** indicate that the model can run on both CPUs and GPUs. • The default runtime is **python2.7**.
MindSpore	aarch64 (recommended)	AArch64 can run only on Snt3 chips.

12.3 How Does ModelArts Use Tags to Manage Resources by Group?

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

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

Operation Process

1. **Step 1 Create Predefined Tags on TMS**
2. **Step 2 Add a Tag to a ModelArts Task**
3. **Step 3 Obtain ModelArts Resource Usage by Resource Type in TMS**

Step 1 Create Predefined Tags on TMS

Log in to the TMS console and create tags on the **Predefined Tags** page. The created tags are global and can be used in all Huawei Cloud regions.

Step 2 Add a Tag to a ModelArts Task

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

- Add a tag to a ModelArts notebook instance.

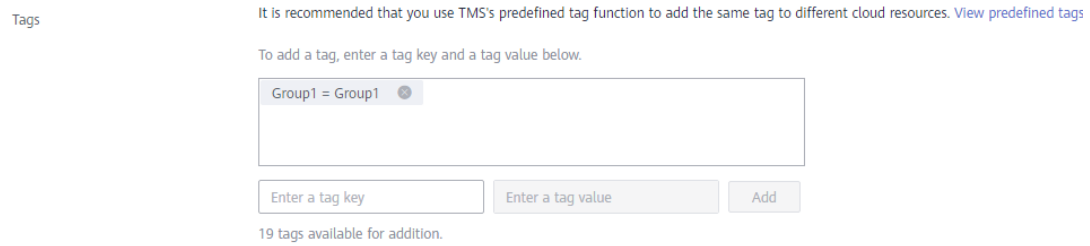
  Add a tag when you create a notebook instance. Alternatively, after creating a notebook instance, add a tag on the **Tags** tab on the instance details page.
- Add a tag to a ModelArts training job.

Add a tag when you create a training job. Alternatively, after creating a training job, add a tag on the **Tags** tab on the job details page.

- Add a tag to a ModelArts real-time service.

  Add a tag when you create a real-time service. Alternatively, after creating a real-time service, add a tag on the **Tags** tab on the service details page.

Figure 12-1 Adding a tag



NOTE

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

Step 3 Obtain ModelArts Resource Usage by Resource Type in TMS

Log in to the TMS console. On the **Resources Tag** page, view resource tasks in specified regions based on resource types and tags.

- **Region**: one or more Huawei Cloud regions.
- **Resource Type**: Table 12-4 lists the resource types that can be viewed on ModelArts.
- **Resource Tag**: If no tag is specified, all resources are displayed, regardless of whether the resources are configured with tags. One or multiple tags can be selected to obtain resource usage.

Table 12-4 Resource types that can be viewed on ModelArts

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

NOTE

If your organization has configured tag policies for ModelArts, add tags to resources based on the policies. If a tag does not comply with the tag policies, resource creation may fail. Contact your organization administrator to learn more about tag policies.

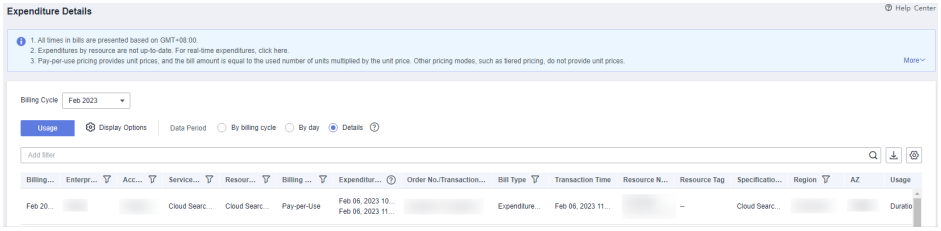
12.4 How Do I View ModelArts Expenditure Details?

In **Billing Center**, the expenditure details of ModelArts are displayed on an hourly basis. You can switch to the **Expenditure Details** page to view the fees consumed for each job.

Procedure:

1. In **Billing Center**, choose **Billing** > **Expenditure Items**. On the **Expenditure Items** page, click the order or transaction number of a record in the list to view its expenditure details.
2. On the **Expenditure Details** page, the resource usage and fees in the order are displayed. Select display options and data period to view expenditure details.

Figure 12-2 Expenditure details



12.5 What Do I Do If the VS Code Window Is Not Displayed?

Possible Cause

VS Code is not installed or the installed version is outdated.

Solution

Download and install VS Code. (Windows users click **Windows**. Users of other operating systems click **another OS**.) After the installation, click **refresh** to complete the connection.

12.6 What Do I Do If a Remote Connection Failed After VS Code Is Opened?

NOTICE

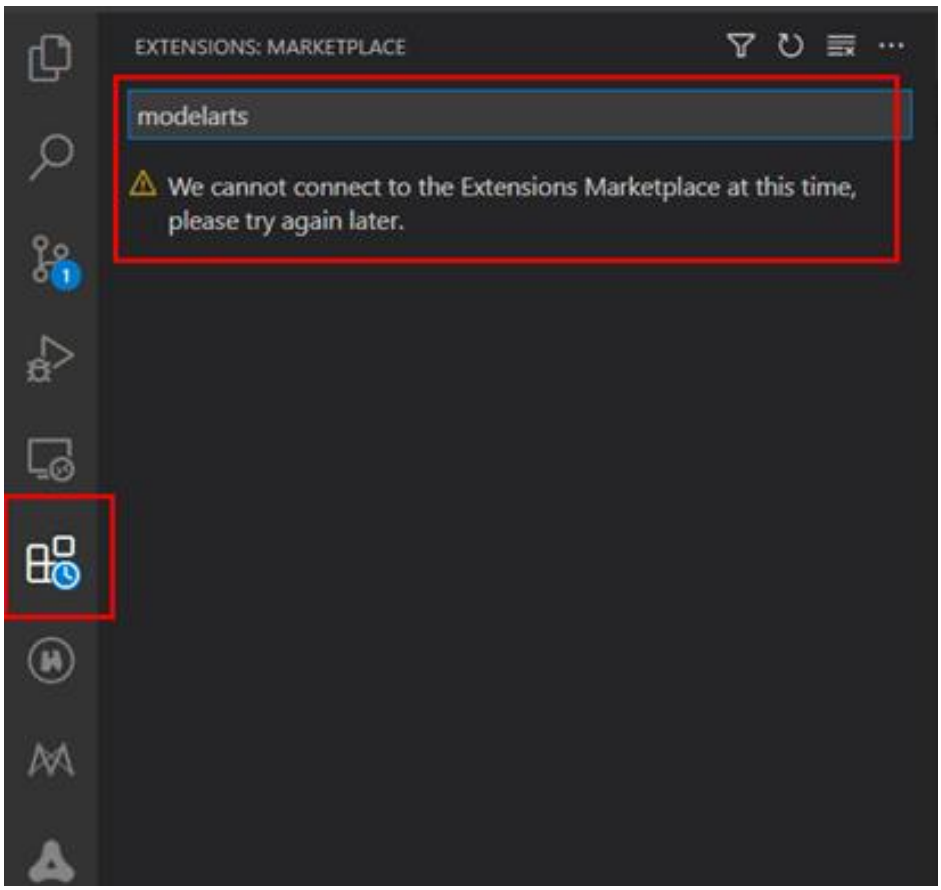
If your local PC runs Linux, see possible cause 2.

Possible Cause 1

Automatically installing the VS Code plug-in ModelArts-HuaweiCloud failed.

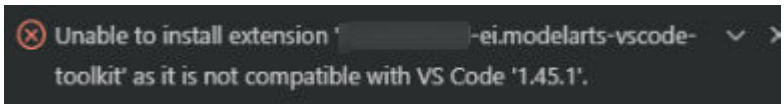
Solution 1

Method 1: Verify that the VS Code network is accessible. Search for **ModelArts-HuaweiCloud** in the VS Code marketplace. If the following information is displayed, a network error occurred. In this case, switch to another proxy or use another network.



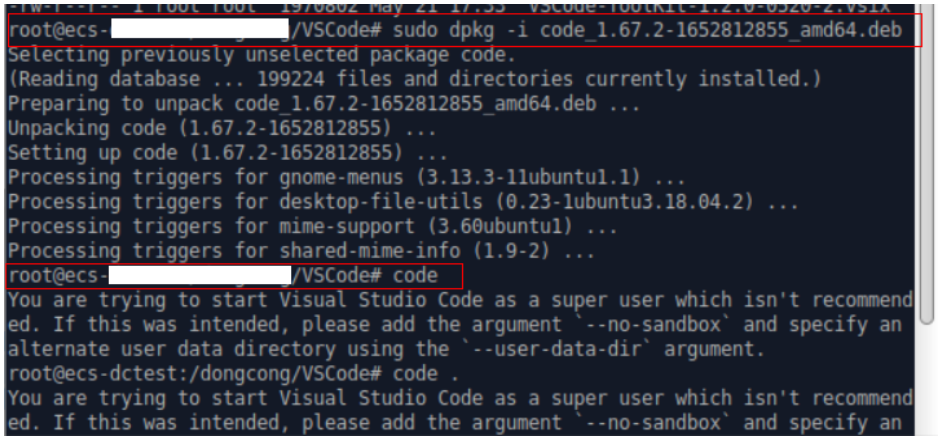
Search for the plug-in again. If the following information is displayed, the network is normal. Then, switch back to the ModelArts console and try to access VS Code again.

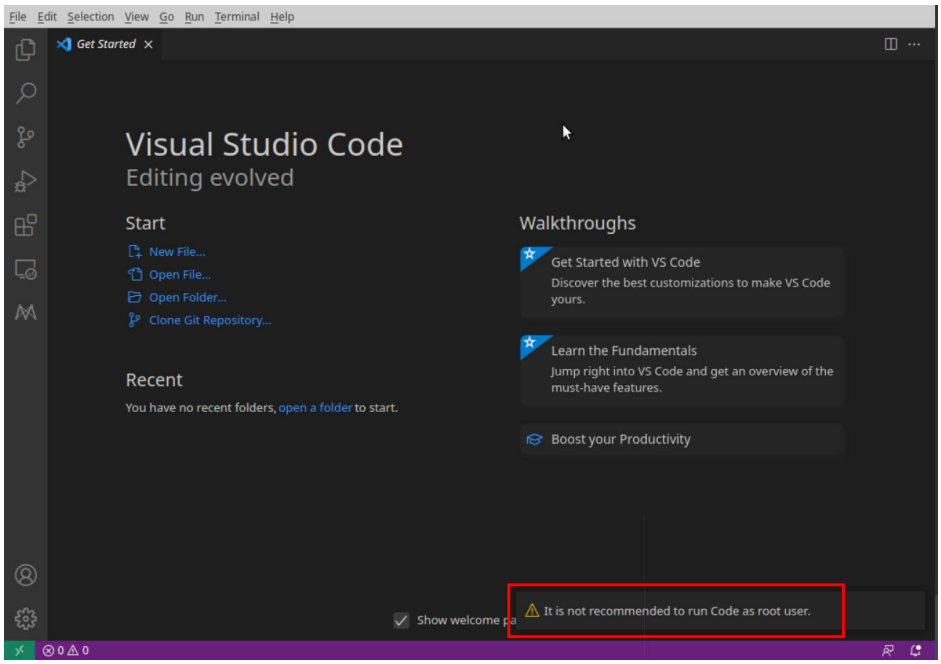
Method 2: If the error message shown in the following figure is displayed, the VS Code version is outdated. Upgrade the VS Code to 1.57.1 or the latest version.



Possible Cause 2

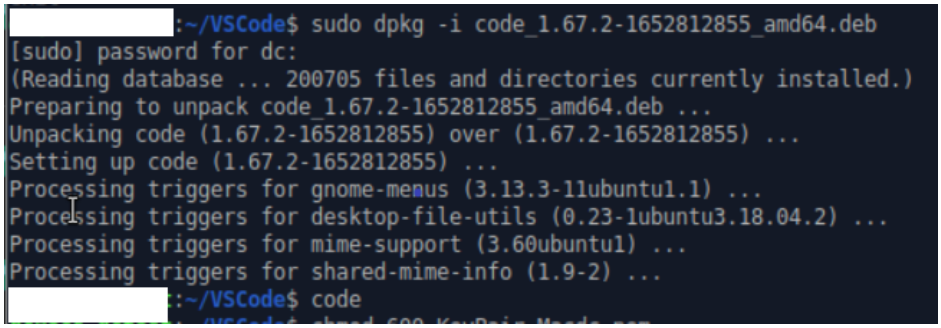
The local PC runs Linux, and VS Code is installed as user **root**. When you access VS Code, the information "It is not recommended to run Code as root user" is displayed.

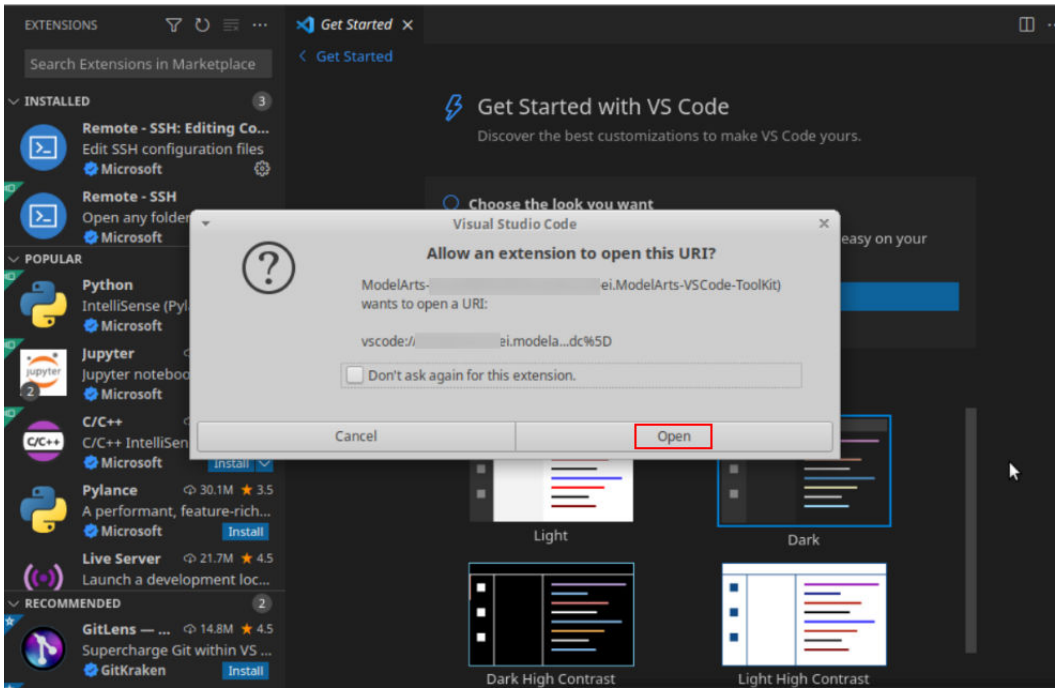




Solution 2

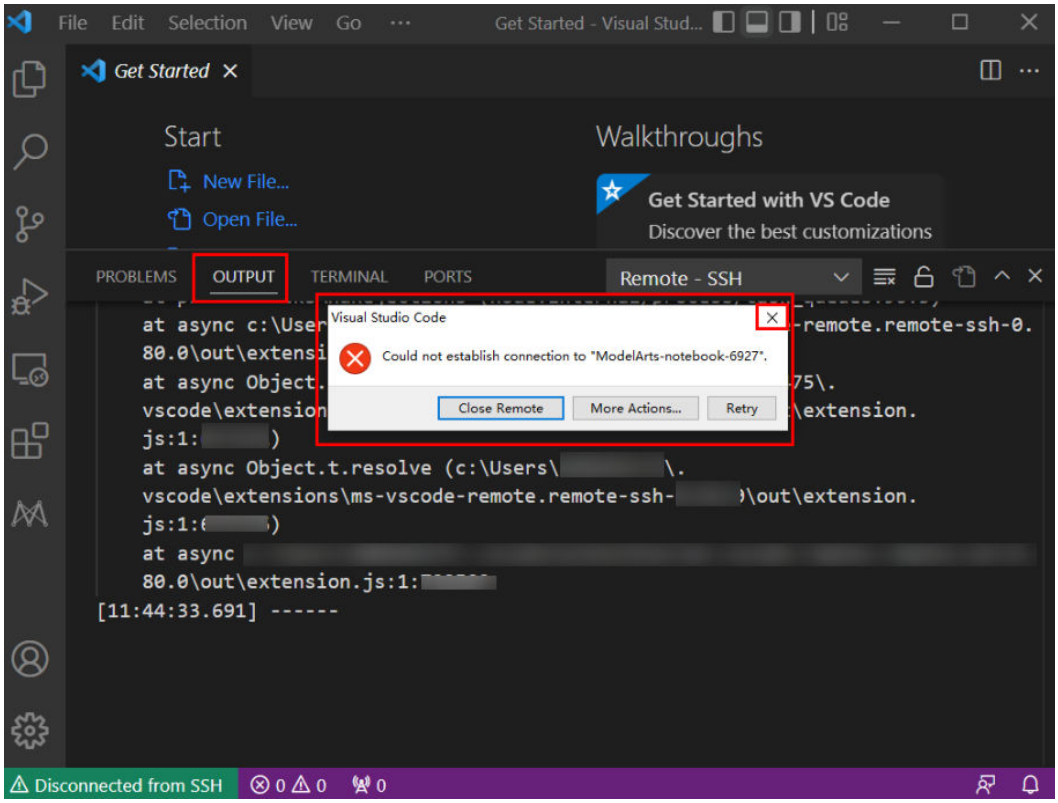
Install VS Code as a non-**root** user, return to the ModelArts management console, and click **Access VS Code**.





12.7 What Do I Do If Error Message "Could not establish connection to xxx" Is Displayed During a Remote Connection?

Symptom



Possible Cause

Establishing a remote SSH connection to an instance through VS Code failed.

Solution

Close the displayed dialog box, view the error information in **OUTPUT**, and rectify the fault by referring to the troubleshooting methods provided in the following sections.

12.8 What Do I Do If Error Message "Bad owner or permissions on C:\Users\Administrator/.ssh/config" or "Connection permission denied (publickey)" Is Displayed?

Symptom

The following error message is displayed: "Bad owner or permissions on C:\Users \Administrator/.ssh/config" or "Connection permission denied (publickey). Please make sure the key file is correctly selected and the file permission is correct. You can view the instance keypair information on ModelArts console."

Possible Causes

The permission to the SSH folder has been granted to other users, not only to the current Windows user, or the current user does not have the permission. In these cases, you only need to modify the permission.

Solution

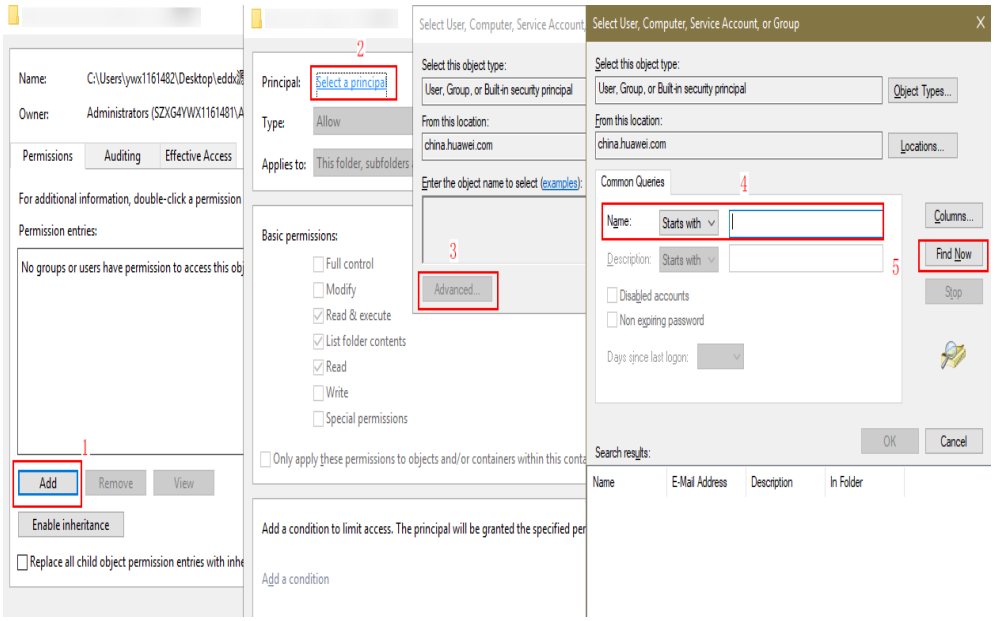
1. Find the SSH folder, which is typically located in **C:\Users**, for example, **C:\Users\xxx**.

   NOTE

   The file name in **C:\Users** must be the same as the Windows login username.

2. Right-click the folder and choose **Properties**. Then, click the **Security** tab.
3. Click **Advanced**. In the displayed window, click **Disable inheritance**. Then, in the **Block Inheritance** dialog box, click **Remove all inherited permissions from this object**. In this case, all users will be deleted.
4. Add an owner. In the same window, click **Add**. In the displayed window, click **Select a principal** next to **Principal**. In the displayed **Select User, Computer, Service Account, or Group** dialog box, click **Advanced**, enter the username, and click **Find Now**. Then, the search results will be displayed. Select your account and click **OK** to close all windows.

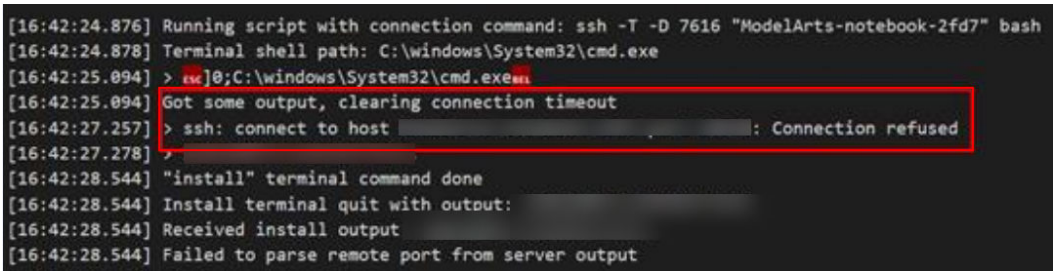
Figure 12-3 Adding an owner



5. Close and open VS Code again and try to remotely access the SSH host. Ensure that the target key is stored in the SSH folder.

12.9 What Do I Do If Error Message "ssh: connect to host xxx.pem port xxxxx: Connection refused" Is Displayed?

Symptom



Possible Cause

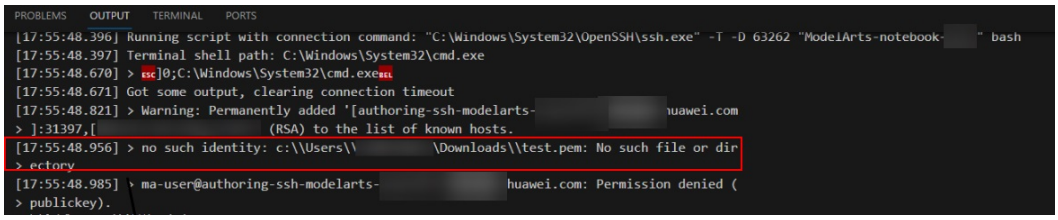
The target instance is not running.

Solution

Log in to the ModelArts management console and check the status of the instance. If the instance is stopped, start it. If the instance is in other states, such as **Error**, stop and then start it. After the instance status changes to **Running**, establish the remote connection again.

12.10 What Do I Do If Error Message "no such identity: C:/Users/xx /test.pem: No such file or directory" Is Displayed?

Symptom



Possible Cause

The key file is not in the path, or the name of the key file in the path has been changed.

Solution

Select the key path again.

12.11 What Are the Precautions for Switching Training Jobs from the Old Version to the New Version?

The differences between the new version and the old version lie in:

- **Differences in Training Job Creation**
- **Differences in Training Code Adaptation**
- **Differences in Built-in Training Engines**

Differences in Training Job Creation

- In earlier versions, you can create a training job using **Algorithm Management**, **Frequently-used**, and **Custom**.
- In the new version, you can create a training job using **Custom algorithm**or **My algorithm**.

This allows you to select algorithms by category.

- The saved algorithms in **Algorithm Management** in the old version are in **My algorithm** in the new version.
- The **Frequently-used** in the old version is the **Custom algorithm** in the new version. Select **Preset image** for **Boot Mode** when you create jobs using the new version.
- The **Custom** in the old version is the **Custom algorithm** in the new version. Select **Custom image** for **Boot Mode** when you create jobs using the new version.

Differences in Training Code Adaptation

In the old version, you are required to configure data input and output as follows:

```
# Parse CLI parameters.
import argparse
parser = argparse.ArgumentParser(description='MindSpore Lenet Example')
parser.add_argument('--data_url', type=str, default="./Data",
                    help='path where the dataset is saved')
parser.add_argument('--train_url', type=str, default="./Model", help='if is test, must provide\
                    path where the trained ckpt file')
args = parser.parse_args()
...
# Download data to your local container. In the code, local_data_path specifies the training input path.
mox.file.copy_parallel(args.data_url, local_data_path)
...
# Upload the local container data to the OBS path.
mox.file.copy_parallel(local_output_path, args.train_url)
```

In the new version, you only need to configure training input and output. In the code, **arg.data_url** and **arg.train_url** are used as local paths. For details, see **Developing Code for Training Using a Preset Image**.

```
# Parse CLI parameters.
import argparse
parser = argparse.ArgumentParser(description='MindSpore Lenet Example')
parser.add_argument('--data_url', type=str, default="./Data",
                    help='path where the dataset is saved')
parser.add_argument('--train_url', type=str, default="./Model", help='if is test, must provide\
                    path where the trained ckpt file')
args = parser.parse_args()
...
# The downloaded code does not need to be set. Use data_url and train_url for data training and output.
# Download data to your local container. In the code, local_data_path specifies the training input path.
#mox.file.copy_parallel(args.data_url, local_data_path)
...
# Upload the local container data to the OBS path.
#mox.file.copy_parallel(local_output_path, args.train_url)
```

Differences in Built-in Training Engines

- In the new version, MoXing 2.0.0 or later is installed by default for built-in training engines.
- In the new version, Python 3.7 or later is used for built-in training engines.
- In the new image, the default home directory has been changed from **/home/work** to **/home/ma-user**. Check whether the training code contains hard coding of **/home/work**.
- Built-in training engines are different between the old and new versions. Commonly used built-in training engines have been upgraded in the new version.

  To use a training engine in the old version, switch to the old version. **Table 12-5** lists the differences between the built-in training engines in the old and new versions.

Table 12-5 Differences between the built-in training engines in the old and new versions

Runtime Environment	Built-in Training Engine and Version	Old Version	New Version
TensorFlow	TensorFlow-1.8.0	√	x
	TensorFlow-1.13.1	√	Coming soon
	TensorFlow-2.1.0	√	√
MXNet	MXNet-1.2.1	√	x
Caffe	Caffe-1.0.0	√	x
Spark MLlib	Spark-2.3.2	√	x
Ray	Ray-0.7.4	√	x
XGBoost with scikit-learn	XGBoost-0.80-Sklearn-0.18.1	√	x
PyTorch	PyTorch-1.0.0	√	x
	PyTorch-1.3.0	√	x
	PyTorch-1.4.0	√	x
	PyTorch-1.8.0	x	√
MPI	MindSpore-1.3.0	x	√
Horovod	Horovod_0.20.0-TensorFlow_2.1.0	x	√
	horovod_0.22.1-pytorch_1.8.0	x	√
MindSpore-GPU	MindSpore-1.1.0	√	x
	MindSpore-1.2.0	√	x

