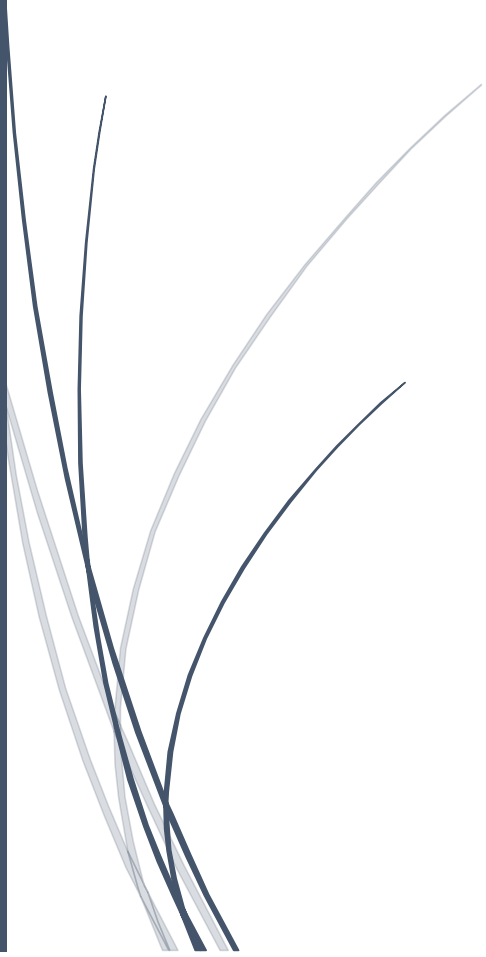
11/10/2020

HUAWEI CLOUD Hands-on lab

Will show how to use MoXing to recognize handwritten digits and images from an MNIST dataset on the Modelarts platform.



HUAWEI CLOUD

Hands-on lab ---Modelarts

Concept Introduction

This section describes how to use MoXing to recognize handwritten digits and images from an MNIST dataset on the ModelArts platform.

MoXing: MoXing is the network model development API provided by the HUAWEI CLOUD deep learning service. Compared with native APIs such as TensorFlow and MXNet, MoXing API simplifies code writing for models. Users only need to care about data input (input_fn) and model build (model_fn) code to implement high-performance running of any model in multiple GPUs and distributed systems.

Overview

The following figure shows the process of identifying handwritten digits and images using MoXing.

1. Region Recommendation: **Singapore**
2. Preparing Data: Obtain the MNIST dataset and upload it to OBS.
3. Training a Model: Use the MoXing framework to compile the model training script and create a training job for model training.
4. Deploying the Model: After obtaining the trained model file, create a prediction job to deploy the model as a real-time prediction service.
5. Verifying the Model: Initiate a prediction request and obtain the prediction result.

Preparing Data

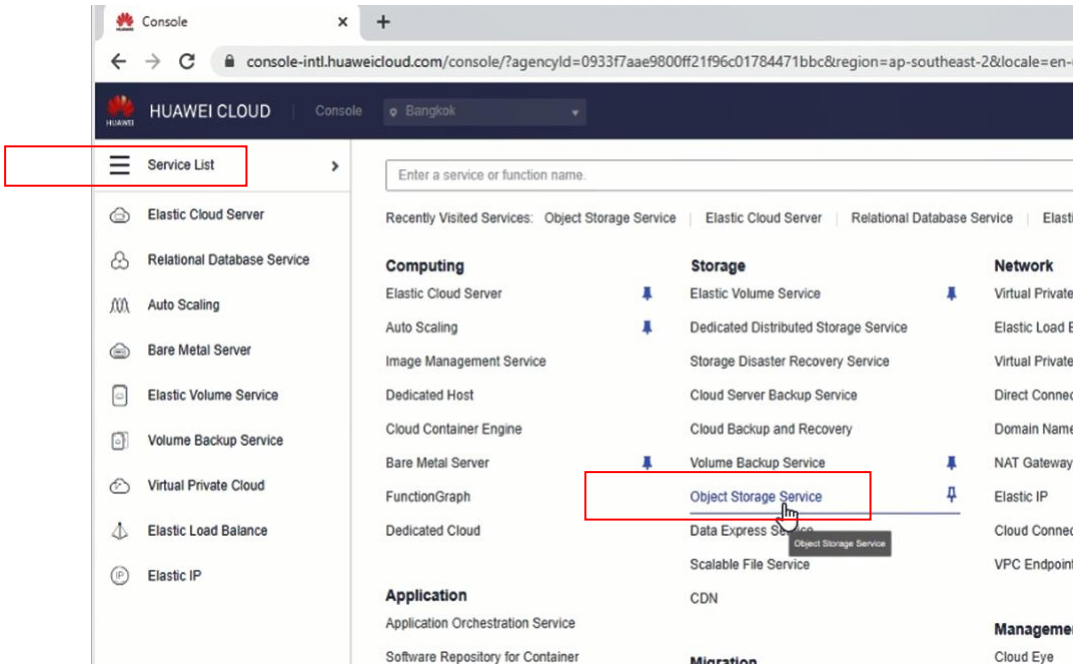
ModelArts provides a sample MNIST dataset named **Mnist-Data-Set**. This example uses this dataset to build a model. Perform the following operations to upload the dataset to the OBS directory **test-modelarts/dataset-mnist** created in preparation.

1. Decompress the **Mnist-Data-Set.zip** file, for example, to the **Mnist-Data-Set** directory on the local PC.

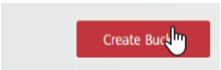
Mnist-Data-Set download link: https://test-modelarts-hol001.obs.ap-southeast-1.myhuaweicloud.com/Modelars-demo-sample/Mnist-Data-Set.zip

2. Create the OBS bucket "**test-modelarts/dataset-mnist**"

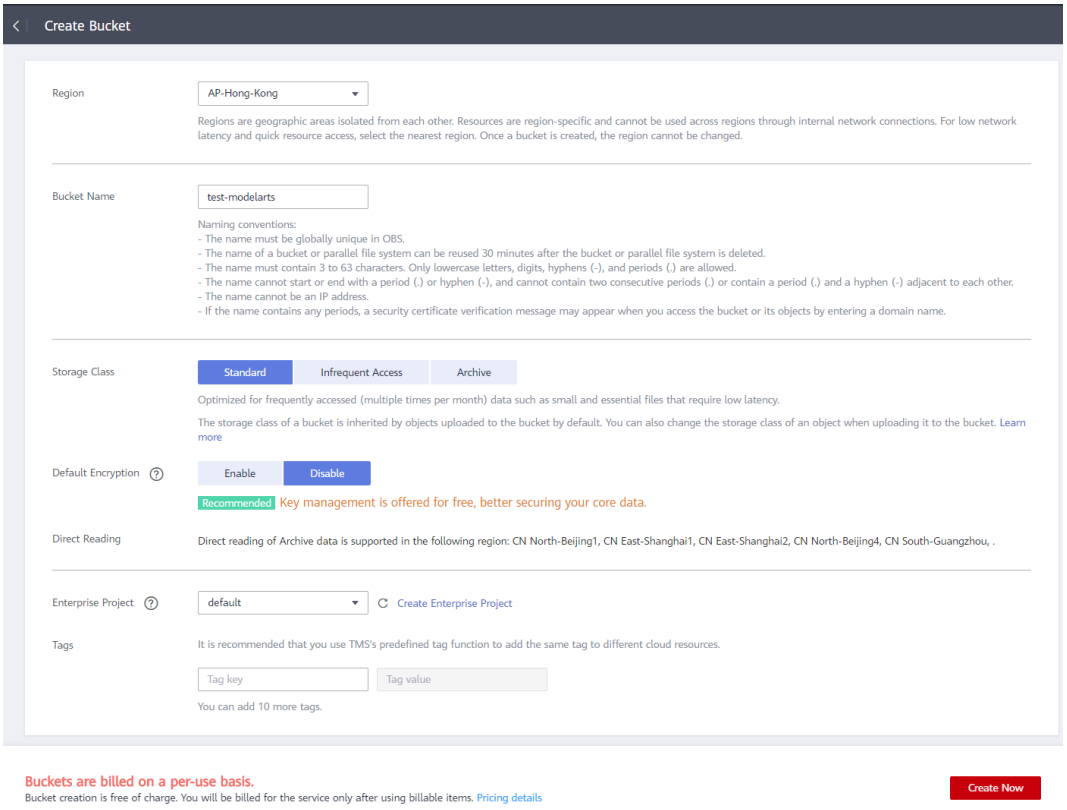
   In the Console, on the top left of the screen, select Navigation menu > Storage > Object Storage Service



   Click on Create Bucket

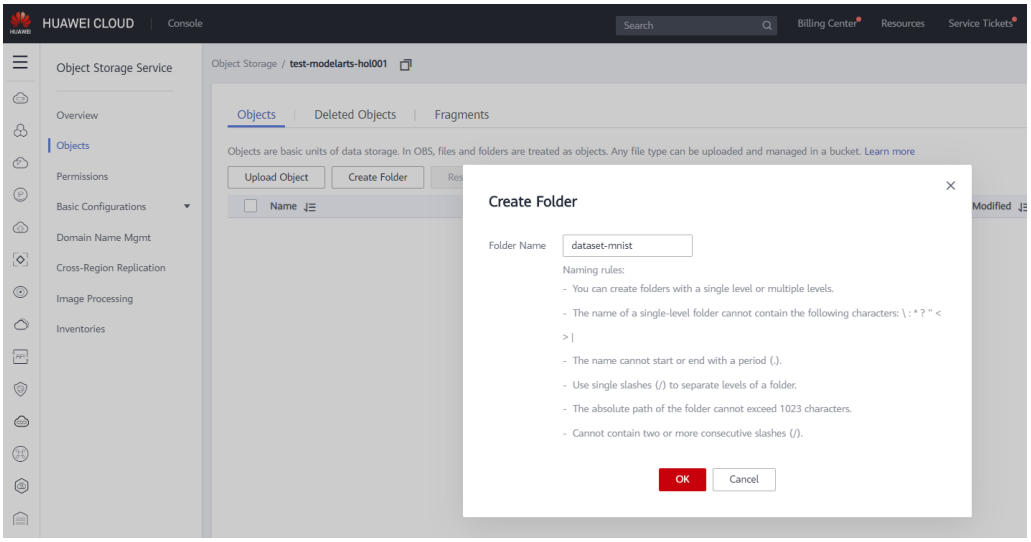


   Create the bucket test-modelars



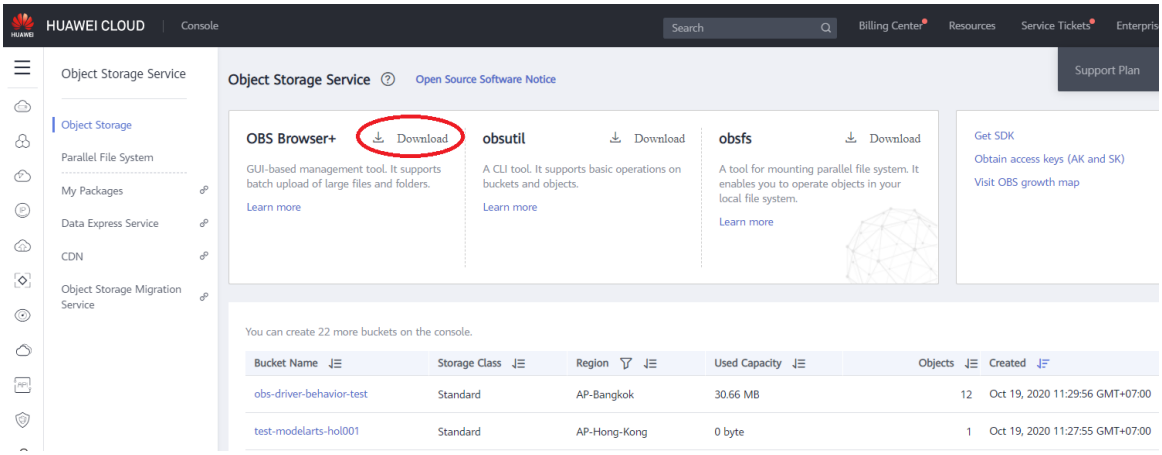
If the console informs that '<u>The bucket name already exists or used by other users. Try another one.</u>' Suggest to using other bucket name such as **test-modelars-hol001**

Create the folder **dataset-mnist** in the bucket test-modelars-hol001

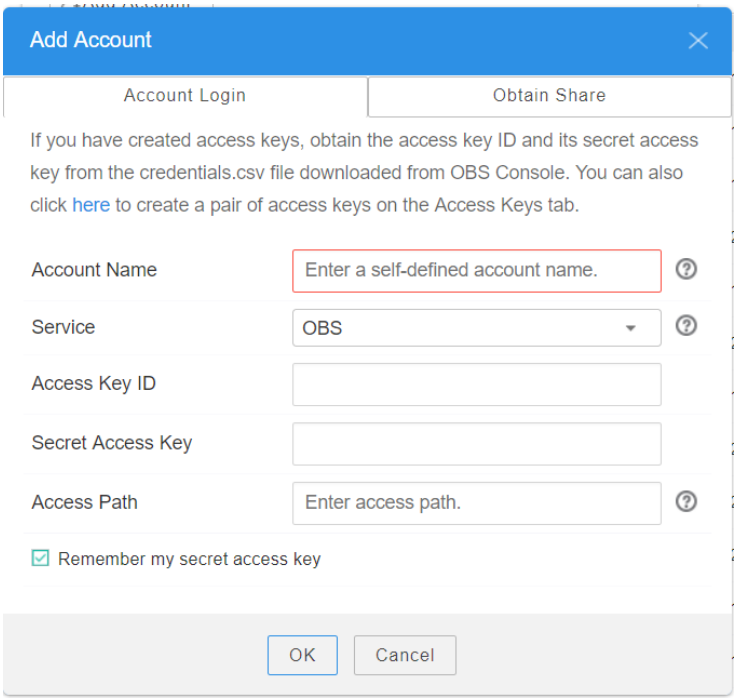


3. Upload all files in the **Mnist-Data-Set** folder to the **test-modelarts/dataset-mnist** directory on OBS in batches. For details about how to upload files, see as follows:

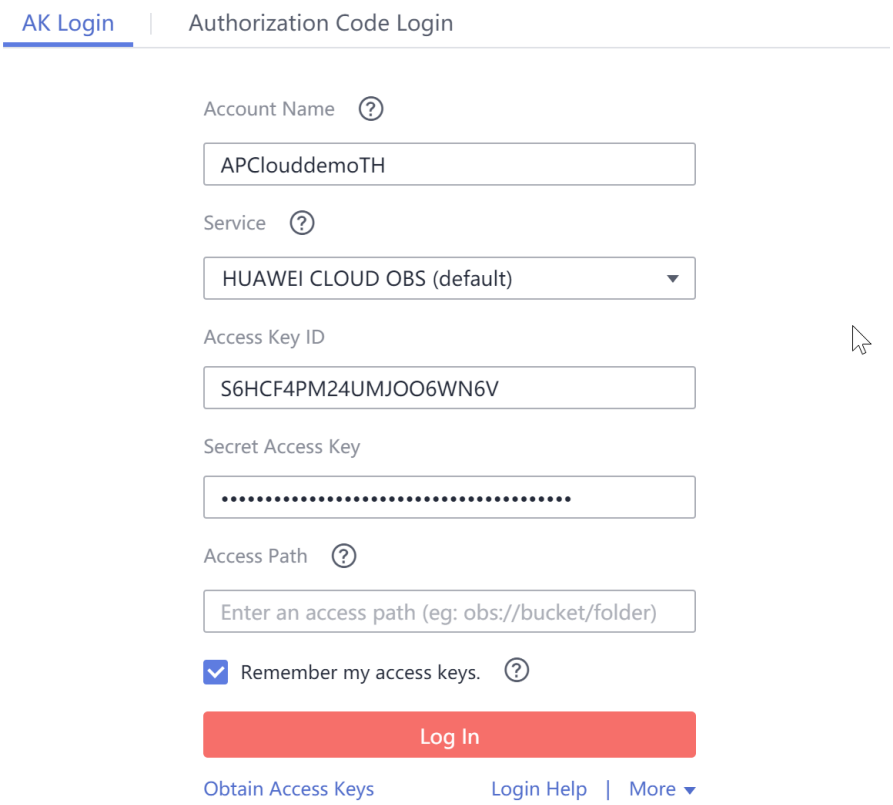
Download the OBS browser

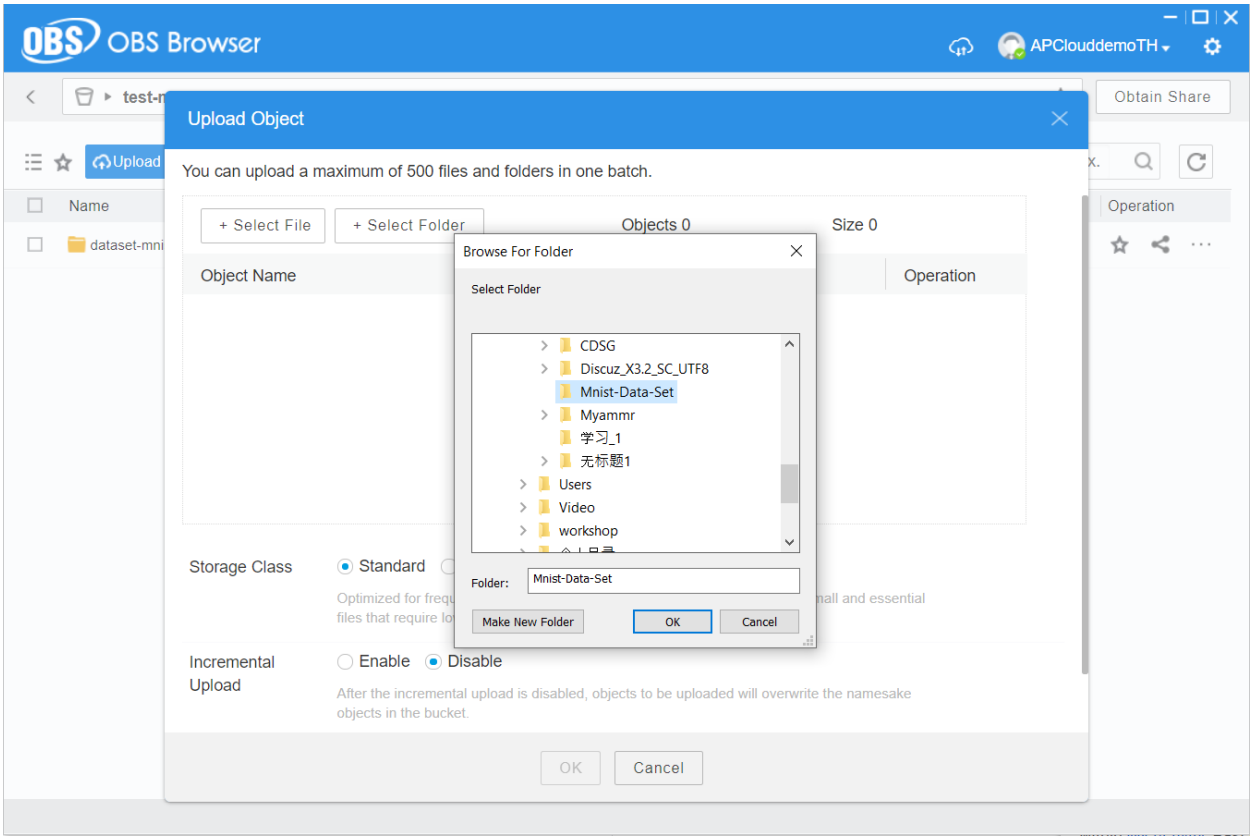


Configure the OBS browser, Account Name, Access Key ID (AK), Secret Access Key (SK) is mandatory.



Upload the file directory to cloud OBS bucket.


AK Login
Authorization Code Login
Account Name
APClouddemoTH
Service
HUAWEI CLOUD OBS (default)
Access Key ID
S6HCF4PM24UMJOO6WN6V
Secret Access Key
Access Path
Enter an access path (eg: obs://bucket/folder)
Remember my access keys.
Log In
Obtain Access Keys
Login Help
More



The following provides content of the **Mnist-Data-Set** dataset. **.gz** is the compressed package.

- **t10k-images-idx3-ubyte**: validation set, which contains 10,000 samples
- **t10k-images-idx3-ubyte.gz**: compressed package file of the validation set.
- **t10k-labels-idx1-ubyte**: labels of the validation set, which contains the labels of the 10,000 samples
- **t10k-labels-idx1-ubyte.gz**: compressed package file of the validation set label.
- **train-images-idx3-ubyte**: training set, which contains 60,000 samples
- **train-images-idx3-ubyte.gz**: compressed package file of the training set.
- **train-labels-idx1-ubyte**: labels of the training set, which contains the labels of the 60,000 samples
- **train-labels-idx1-ubyte.gz**: compressed package file of the training set label.

Training a Model

After the data preparation is completed, use the MoXing API to compile the training script code. ModelArts provides a code sample, **train_mnist.py**. The following uses this sample to train the model.

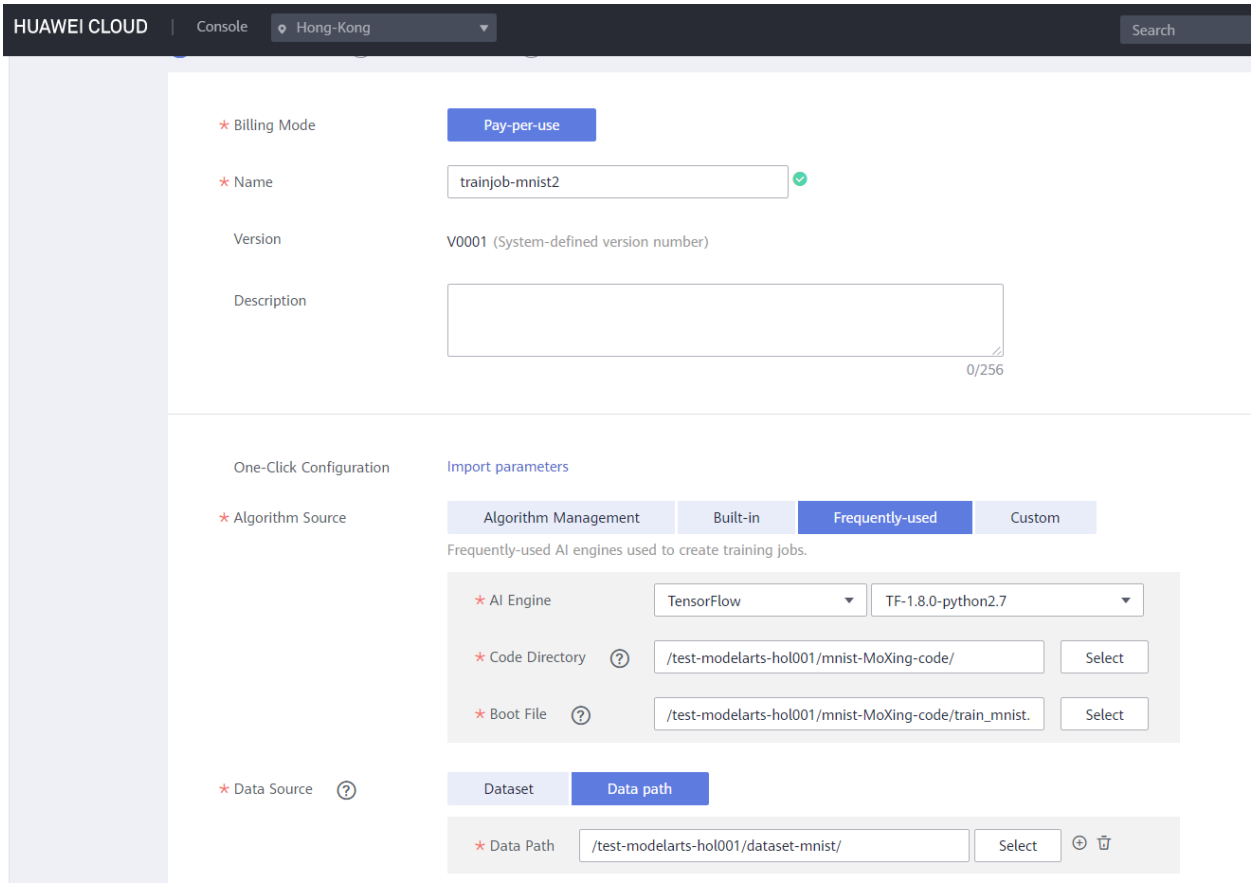
1. Upload the **train_mnist.py** file to OBS, for example, **test-modelarts-hol001/mnist-MoXing-code**.

   Down load link: https://test-modelarts-hol001.obs.ap-southeast-1.myhuaweicloud.com/Modelars-demo-sample/train_mnist.py

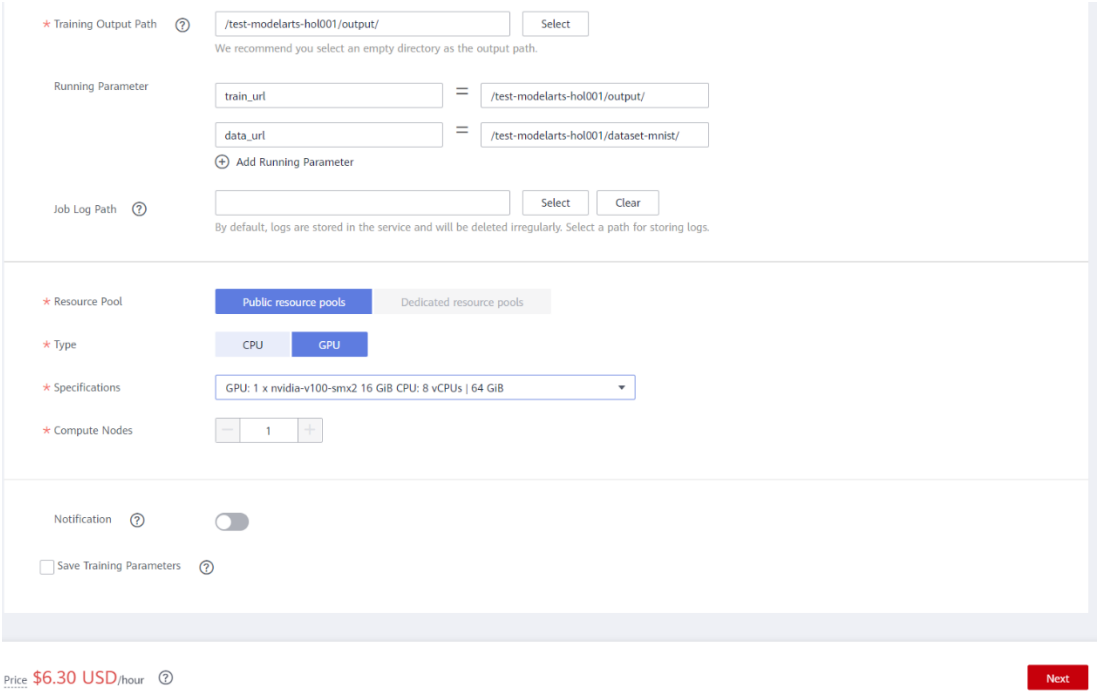
2. On the ModelArts management console, choose **Training Management > Training Jobs**, and click **Create** in the upper left corner.
3. On the **Modelarts Console** page, click **Training Management->Training Jobs** and Click **Create**.

   Data Source: Select **Data path**, and then select the OBS path for saving the dataset.

   Basic information for creating a training job



Parameters for creating a training job



4. On the **Confirm** tab page, check the parameters of the training job and click **Submit**.
5. On the **Training Jobs** page, when the training job status changes to **Running Success**, the model training is completed. If any exception occurs, click the job name to go to the job details page and view the training job logs.

> NOTE:
>
> The training job may take more than 10 minutes to complete. If the training time exceeds a certain period (for example, one hour), manually stop it to release resources. Otherwise, the account balance may be insufficient, especially for the models that are trained using GPUs.

6. (Optional) During or after model training, you can create a visualization job to view parameter statistics.

   In **Training Output Path**, select the value of **Training Output Path** specified for the training job. Complete visualization job creation as prompted.

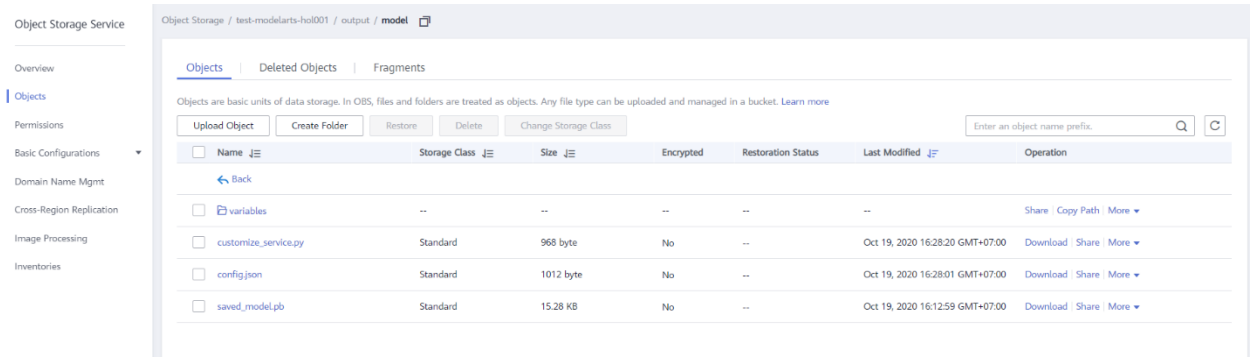
Deploying the Model

After the model training is completed, deploy the model as a real-time prediction service. ModelArts provides the compiled inference code **customize_service.py** and configuration file **config.json**.

Customize_service.py download link: https://test-modelarts-hol001.obs.ap-southeast-1.myhuaweicloud.com/Modelars-demo-sample/customize_service.py

Config.json download link: https://test-modelarts-hol001.obs.ap-southeast-1.myhuaweicloud.com/Modelars-demo-sample/config.json

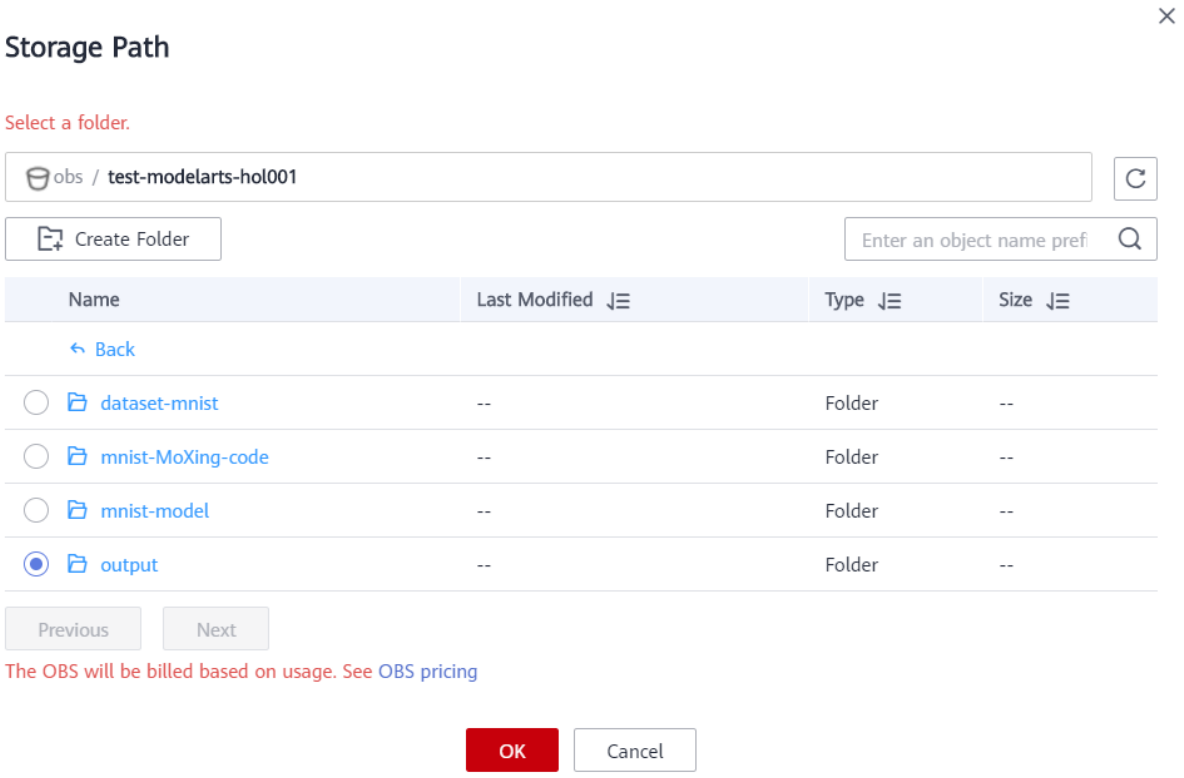
1. Upload the **customize_service.py** and **config.json** files to OBS. The files must be stored in the path for saving the model generated for the training job, for example, **test-modelarts-hol001/output/model**.



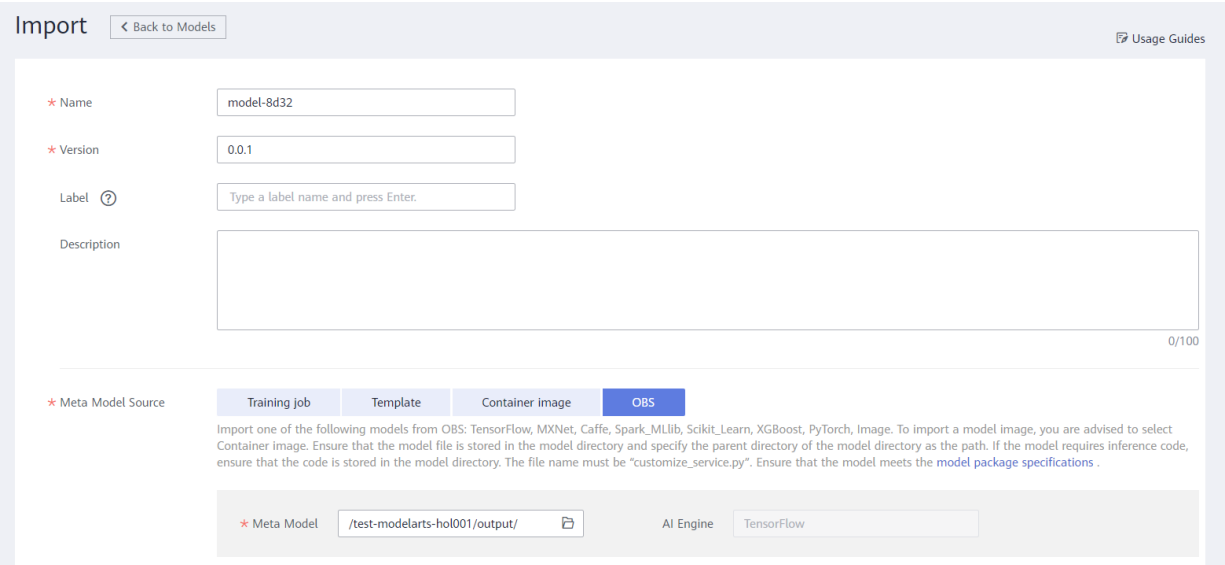
NOTE:

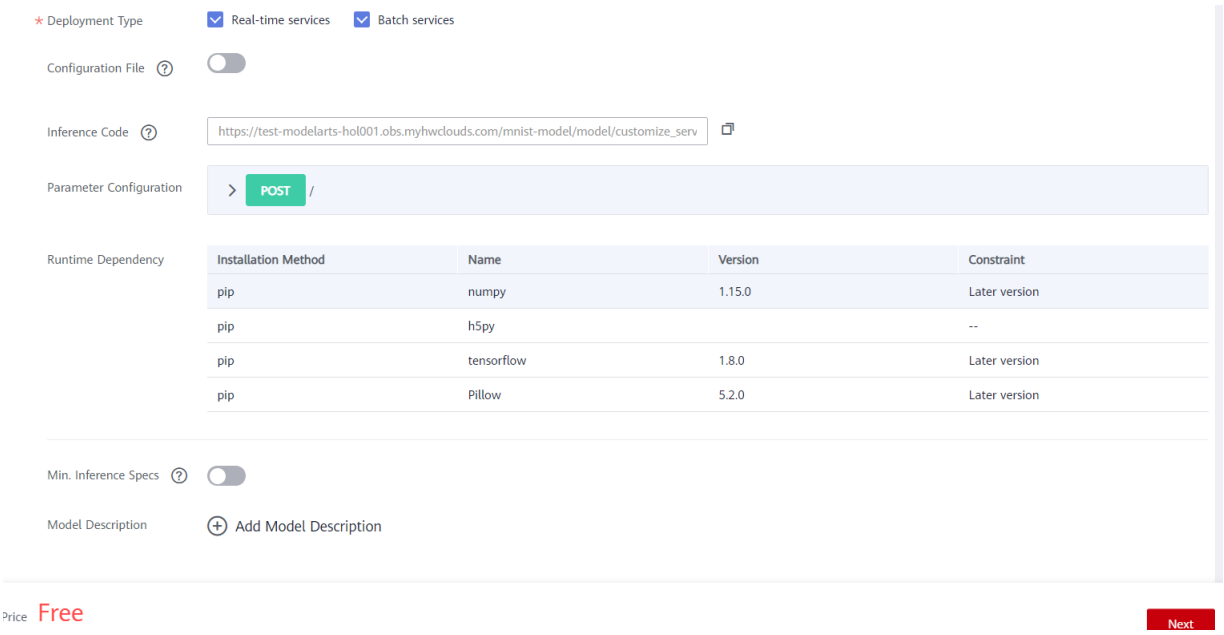
- The training job creates a **model** folder in the path specified by **Training Output Path** to store the generated model.
- The inference code and configuration file must be uploaded to the **model** folder.

2. On the ModelArts management console, choose **Model Management** > **Models** in the left navigation pane. The **Models** page is displayed. Click **Import** in the upper left corner.
3. On the **Import Model** page, set required parameters as shown and click **Next**.
   In **Meta Model Source**, select **OBS**. Set **Meta Model** to the path specified by **Training Output Path** in the training job but not the **model** folder under the path. Otherwise, the system cannot find the model and related files automatically.



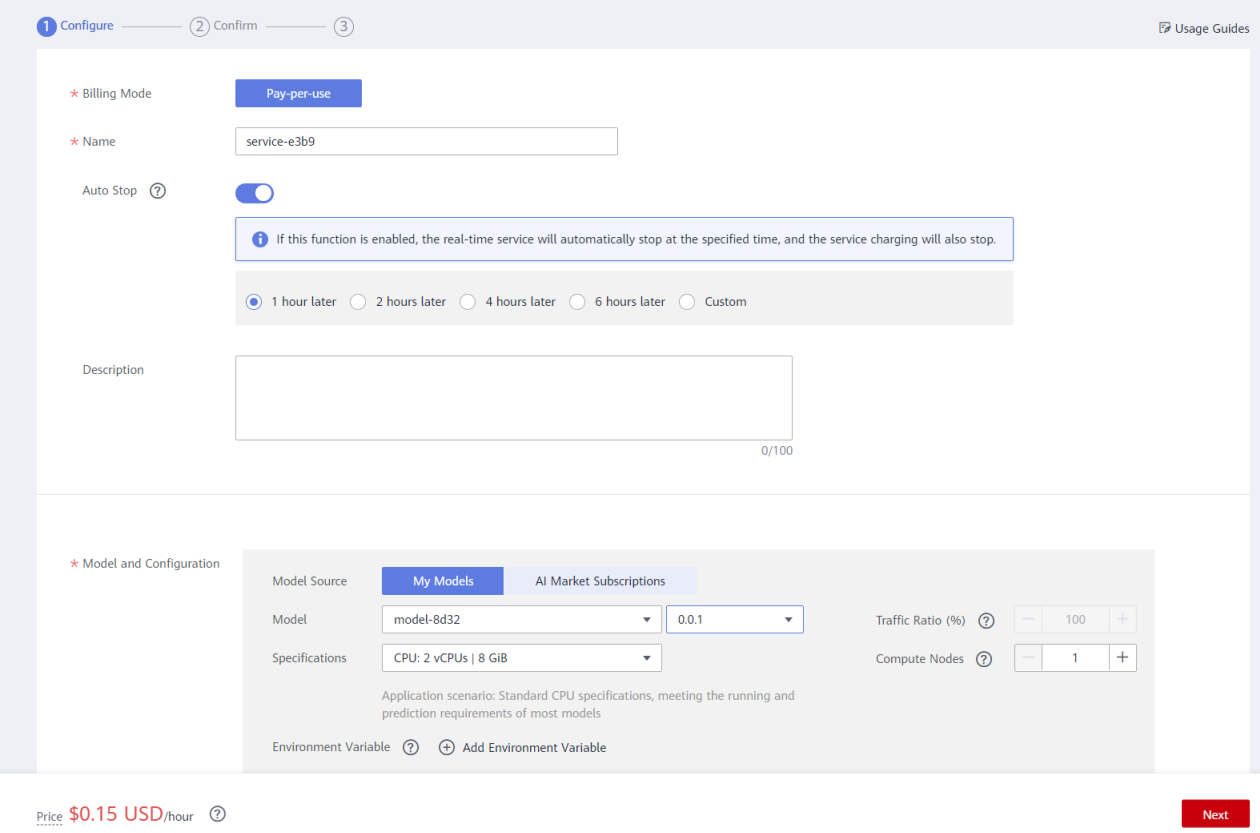
Import Model





4. On the **Models** page, if the model status changes to **Normal**, the model has been imported successfully. Click the triangle next to a model name to expend all versions of the model. In the row of a version, choose **Deploy** > **Real-Time Services** in the **Operation** column to deploy the model as a real-time service.
5. On the **Deploy** page, set parameters by referring to Figure 4 and click **Next**.

Figure 4 Deploy



6. On the **Confirm** tab page, check the configurations and click **Submit** to create a real-time service.
7. After the real-time service is created, the **Service Deployment** > **Real-Time Services** page is displayed. The service deployment takes some time. When the service status changes to **Running**, the service is successfully deployed.

Verifying the Model

After the real-time service is deployed, access the service to send a prediction request for test.

1. On the **Real-Time Services** page, click the name of the real-time service. The real-time service details page is displayed.
2. On the real-time service details page, click the **Prediction** tab.
3. Click **Upload** next to **Image File** to upload an image with a white handwritten digit on a black background and click **Predict**.

   Testing data download link: https://test-modelarts-hol001.obs.ap-southeast-1.myhuaweicloud.com/Modelars-demo-sample/test%20data.zip

After the prediction is completed, the prediction result is displayed in the **Test Result** pane. According to the prediction result, the digit on the image is **4**.

NOTE:

- As specified in the inference code and configuration files, the size of the image used for prediction must be 28 x 28 pixels, and the image must be in the JPG format and must contain white handwritten digits on a black background.
- You are advised not to use the images provided by the dataset. You can use the drawing tool provided by the Windows operating system to draw an image for prediction.
- If a single-channel image that is not in the required format is used, the prediction result may be inaccurate.

Figure 5 Prediction result of the real-time service