ModelArts

Development Environment

 Issue
 01

 Date
 2023-11-22





Copyright © Huawei Technologies Co., Ltd. 2023. All rights reserved.

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

Trademarks and Permissions

NUAWEI and other Huawei trademarks are trademarks 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 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 Technologies Co., Ltd.

- Address: Huawei Industrial Base Bantian, Longgang Shenzhen 518129 People's Republic of China Website: https://www.huawei.com
- Email: <u>support@huawei.com</u>

Security Declaration

Vulnerability

Huawei's regulations on product vulnerability management are subject to "Vul. Response Process". For details about the policy, see the following website:<u>https://www.huawei.com/en/psirt/vul-response-process</u> For enterprise customers who need to obtain vulnerability information, visit:<u>https://securitybulletin.huawei.com/enterprise/en/security-advisory</u>

Contents

1 Introduction to DevEnviron	1
2 Application Scenarios	5
3 Managing Notebook Instances	6
3.1 Creating a Notebook Instance	6
3.2 Accessing a Notebook Instance	10
3.3 Starting, Stopping, or Deleting a Notebook Instance	11
3.4 Changing a Notebook Instance Image	12
3.5 Dynamically Expanding EVS Disk Capacity	12
3.6 Changing the Flavor of a Notebook Instance	13
3.7 Modifying the SSH Configuration for Notebook	14
3.8 Viewing All Notebook Instances of an IAM Project	15
4 JupyterLab	17
4.1 Operation Process in JupyterLab	17
4.2 JupyterLab Overview and Common Operations	18
4.3 JupyterLab Plug-ins	26
4.3.1 Code Parametrization Plug-in	26
4.4 Using ModelArts SDK	29
4.5 Using the Git Plug-in	29
5 Local IDE	35
5.1 Operation Process in a Local IDE	35
5.2 Local IDE (PyCharm)	36
5.2.1 Configuring a Local IDE Accessed Using PyCharm Toolkit	36
5.2.2 Configuring a Local IDE Manually Accessed Using PyCharm	42
5.3 Local IDE (VS Code)	48
5.3.1 Connecting to a Notebook Instance Through VS Code	48
5.3.2 Installing VS Code	48
5.3.3 Connecting to a Notebook Instance Through VS Code with One Click	49
5.3.4 Connecting to a Notebook Instance Through VS Code Toolkit	54
5.3.5 Manually Connecting to a Notebook Instance Through VS Code	63
5.3.6 Remotely Debugging in VS Code	70
5.3.7 Uploading and Downloading a File in VS Code	72
5.4 Configuring a Local IDE Accessed Using SSH	74

6 ModelArts Tool Guide	81
6.1 PyCharm Toolkit	81
6.2 Preparations	
6.2.1 Downloading and Installing PyCharm Toolkit	
6.2.2 Configuring Toolkit Using a YAML File	
6.2.3 Creating Access Keys (AK and SK)	85
6.2.4 Using Access Keys for Login	85
6.3 PyCharm Toolkit (Latest Version)	86
6.3.1 Training a Model	
6.3.1.1 Submitting a Training Job (New Version)	86
6.3.1.2 Stopping a Training Job	90
6.3.1.3 Viewing Training Logs	91
6.4 FAQs	91
6.4.1 What Should I Do If an Error Occurs During ToolKit Installation?	92
6.4.2 An Error Occurs When You Edit a Credential in PyCharm Toolkit	92
6.4.3 Why Cannot I Start Training?	94
6.4.4 What Should I Do If Error "xxx isn't existed in train_version" Occurs When a Training Job Is Submitted	94
6.4.5 What Should I Do If an Error Occurs When I Submit a Training Job	95
6.4.6 What Should I Do If an Error Occurs During Service Deployment	96
6.4.7 How Do I View Error Logs of PyCharm ToolKit?	96
7 Uploading and Downloading Data in Notebook	97
7.1 Uploading Files to JupyterLab	97
7.1.1 Scenarios	97
7.1.2 Uploading Files from a Local Path to JupyterLab	97
7.1.2.1 Upload Scenarios and Entries	
7.1.2.2 Uploading a Local File Less Than 100 MB to JupyterLab	
7.1.2.3 Uploading a Local File with a Size Ranging from 100 MB to 5 GB to JupyterLab	100
7.1.2.4 Uploading a Local File Larger Than 5 GB to JupyterLab	103
7.1.3 Cloning an Open-Source Repository in GitHub	105
7.1.4 Uploading OBS Files to JupyterLab	106
7.1.5 Uploading Remote Files to JupyterLab	108
7.2 Downloading a File from JupyterLab to a Local Path	109
7.3 Uploading Data from a Local IDE to a Notebook Instance	111
7.4 Downloading Files from a Notebook Instance to a Local Directory	112

Introduction to DevEnviron

NOTE

This document describes the functions of new-version DevEnviron notebook. Check the notebook version in the navigation pane on the left. If there is only one **Notebook** item under **DevEnviron**, the notebook is of the new version.

Figure 1-1 New-version notebook



After new-version notebook instances are created, **Notebooks New** will be displayed on the **Dashboard** page.

Dashboard	
Price	
Usage Details	
Notebooks New - [VevEnviron
Active Instances	Projects
1	
	I.

Figure 1-2 Notebooks New on the Dashboard page

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

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

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

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

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

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

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



Figure 1-3 Remotely accessing notebook from a local IDE

Uploaded through web or OBS Browser+

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

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

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

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

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

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

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

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

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

The image versions preset in ModelArts are determined based on user feedback and version stability. If your development can be carried out using the versions preset in ModelArts, for example, MindSpore 1.5, use preset images. These images have been fully verified and have many commonly-used installation packages built in. They are out-of-the-box, relieving you from configuring the environment.

The images preset in ModelArts DevEnviron include:

- Common preset packages: Common AI engines based on standard Conda, common data analysis software packages such as Pandas and Numpy, and common tool software such as CUDA and CUDNN, meet common AI development requirements.
- Preset Conda environments: A Conda environment and basic Conda Python (excluding any AI engine) are created for each preset image. The following figure shows the Conda environment for the preset MindSpore.



Select a Conda environment based on whether the AI engine is used for debugging.

- Notebook: a web application that enables you to code on the GUI and combine the code, mathematical equations, and visualized content into a document.
- JupyterLab plug-ins: enable flavor changing and instance stopping to improving user experience.
- Remote SSH: allows you to remotely debug a notebook instance from a local PC.

- To simplify operations, ModelArts notebook of the new version does not support switchover between AI engines in a notebook instance.
- Al engines vary based on regions. For details about the Al engines available in a region, see the Al engines displayed on the management console.

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

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

JupyterLab is an interactive development environment. It is the next-generation product of Jupyter Notebook. JupyterLab enables you to compile notebooks, operate terminals, edit Markdown text, enable interaction, and view CSV files and images.

2 Application Scenarios

ModelArts provides flexible, open development environments. Select a development environment based on site requirements.

- In-cloud notebook that is out of the box, relieving you from concerning environment installation or configuration. For details, see JupyterLab Overview and Common Operations.
- Local IDE for model development. After enabling remote SSH, you can remotely access the ModelArts notebook development environment to debug and run code from a local IDE. The local IDE allows you to use the in-cloud notebook development environment while with local coding habits unchanged.

The local IDE supports VS Code, PyCharm, and SSH. Additionally, the PyCharm Toolkit and VS Code Toolkit are provided for convenient remote access. For details, see **Connecting to a Notebook Instance Through VS Code with One Click**.

3 Managing Notebook Instances

Creating a Notebook Instance Accessing a Notebook Instance Starting, Stopping, or Deleting a Notebook Instance Changing a Notebook Instance Image Dynamically Expanding EVS Disk Capacity Changing the Flavor of a Notebook Instance Modifying the SSH Configuration for Notebook Viewing All Notebook Instances of an IAM Project

3.1 Creating a Notebook Instance

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

Context

- Only running notebook instances can be accessed or stopped.
- A maximum of 10 notebook instances can be created under an account.

Procedure

1. Log in to the ModelArts management console. In the navigation pane, choose **Settings** and check whether the access authorization has been configured. If not, configure access authorization. For details, see "Configuring Access Authorization".

Figure 3-1 Configuring authorization

Global Configuration ②		
Add Authorization		
Username	Authorization Type	Authorization Content
All users	Agency	modelarts_agency View Permissions

- Log in to the ModelArts management console. In the left navigation pane, choose **DevEnviron > Notebook** to switch to the new-version **Notebook** page.
- 3. Click **Create**. On the **Create Notebook** page, configure parameters.
 - a. Configure basic information of the notebook instance, including its name, description, and auto stop status. For details, see **Table 3-1**.

Figure 3-2 Basic information of a notebook instance

* Name	notebook-ff04	
Description		
	0/256	
* Auto Stop		
	1 Enable this option to specify a time for the notebook instance to automatically stop. You will not be billed after it has stopped.	×
	1 hours 2 Hours 4 Hours 6 Hours Custom	

Table 3-1 Basic parameters

Paramete r	Description
Name	Name of the notebook instance. Enter 1 to 64 characters. Only letters, digits, hyphens (-), and underscores (_) are allowed.
Descriptio n	Brief description of the notebook instance
Auto Stop	Automatically stops the notebook instance at a specified time. This function is enabled by default. The default value is 1 hour , indicating that the notebook instance automatically stops after running for 1 hour.
	The options are 1 hour , 2 hours , 4 hours , 6 hours , and Custom . You can select Custom to specify any integer from 1 to 24 hours.

b. Configure notebook parameters, such as the image and instance flavor. For details, see **Table 3-2**.

Paramete r	Description
Image	Public and private images are supported.
	• Public images are the AI engines built in ModelArts.
	 Private images can be created using an instance that is created using a public image. Custom images have not been officially released.
	An image corresponds to an AI engine. When you select an image during instance creation, the AI engine is specified accordingly. Select an image as required. Enter a keyword of the image name in the search box on the right to quickly search for the image.
	You can change an image on a stopped notebook instance.
Resource Pool	Select a resource pool as required.
Туре	Chip type, which can be CPU or GPU.
	The chips vary depending on the selected image.
Flavor	The flavor of your notebook instance.
Storage	Default, EVS, and SFS can be selected.
	• Default If you select this option, the system provides 50 GB of default free storage for each notebook instance.
	 EVS Set disk space, ranging from 5 GB to 4096 GB, based on actual usage. The default value is 5 GB.
	 SFS, which is supported by dedicated resource pools only
	All the storage paths of Default , EVS , and SFS are mounted to /home/ma-user/work . All read and write operations on files in the notebook instance are stored in this directory, not in OBS.
	The data is retained in /home/ma-user/work , even if the notebook instance is stopped or restarted.
	When the notebook instance is deleted, the data is deleted accordingly.

 Table 3-2 Notebook instance parameters

Paramete r	Description
Remote SSH	 After you enable this function, you can remotely access the development environment of the notebook instance from your local development environment. When a notebook instance is stopped, you can update the SSH configuration on the instance details page. NOTE The notebook instances with remote SSH enabled have VS Code plug-ins (such as Python and Jupyter) and the VS Code server package pre-installed, which occupy about 1 GB persistent storage space.
Key Pair	Set a key pair after remote SSH is enabled.
	Select an existing key pair.
	Alternatively, click Create on the right of the text box to create one on the DEW console. To do so, choose Key Pair Service > Private Key Pairs and click Create Key Pair .
	After a notebook instance is created, you can change the key pair on the instance details page.
	CAUTION Download the created key pair and properly keep it. When you use a local IDE to remotely access the notebook development environment, the key pair is required for authentication.
Whitelist	Set a whitelist after remote SSH is enabled. This parameter is optional.
	Add the IP addresses for remotely accessing the notebook instance to the whitelist, for example, the IP address of your local PC or the public IP address of the source device. A maximum of five IP addresses can be added and separated by commas (,). If the parameter is left blank, all IP addresses will be allowed for remote SSH access.
	If your source device and ModelArts are isolated from each other in network, obtain the public IP address of your source device using a mainstream search engine, for example, by entering "IP address lookup", but not by running ipconfig or ifconfigip locally.
	After a notebook instance is created, you can change the whitelist IP addresses on the instance details page.

- 4. Click Next.
- 5. After confirming the parameter settings, click **Submit**.

Switch to the notebook instance list. The notebook instance is being created. It will take several minutes when its status changes to **Running**. Then, the notebook instance is created.

6. In the notebook instance list, click the instance name. On the instance details page that is displayed, view the instance configuration.

Figure 3-3 Details about a notebook instance

Name Status ID Storage Path Storage Capacity	notecok 2 • Ruming (13 minutes left) () /tome/ma-set/work/ 5 CB (US) Expension	Flavor Image Created At Updated At	CPU 2xCPU 868 * pyterch 1 = cutal 10 2 - cutant2 - disarts 18.04 Dec 66, 2023 08:3131 CMT-98:00 Dec 60, 2023 08:33.95 CMT-98:00
Remote SSH Authentication	negrar osas 🖉	Address	zu husweidoud com

The SSH configuration of a stopped notebook instance can be modified. Both the key and whitelist can be modified.

To modify the whitelist, click the modification icon on the right.

3.2 Accessing a Notebook Instance

Access a notebook instance in the Running state for coding.

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

- Remotely accessed from a local IDE through PyCharm, VS Code, or SSH. For details, see Connecting to a Notebook Instance Through VS Code Toolkit.
- Accessed online using JupyterLab. For details, see JupyterLab Overview and Common Operations.

A ModelArts notebook instance is started as user **ma-user**. The default working directory of the instance is **/home/ma-user**.

sh-4.4\$pwd /home/ma-user sh-4.4\$

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

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

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

3.3 Starting, Stopping, or Deleting a Notebook Instance

Starting or Stopping an Instance

Stop the notebook instances that are not needed. You can also restart a stopped instance.

- Log in to the ModelArts management console. Choose DevEnviron > Notebook in the navigation pane on the left. The notebook list of the new version is displayed.
- 2. Start or stop the target notebook instance.
 - To start a notebook instance, click **Start** in the **Operation** column of the target notebook instance. Only stopped notebook instances can be started.
 - To stop a notebook instance, click **Stop** in the **Operation** column of the target notebook instance. Only running notebook instances can be stopped.

After a notebook instance is stopped:

• The data stored only in **/home/ma-user/work** is retained. For example, the external dependency packages installed in other directories in the development environment will be deleted.

Deleting an Instance

Delete the notebook instances that are not needed.

- Log in to the ModelArts management console. Choose DevEnviron > Notebook in the navigation pane on the left. The notebook list of the new version is displayed.
- 2. In the notebook list, click **Delete** in the **Operation** column of the target notebook instance. In the dialog box that is displayed, click **OK**.

Deleted notebook instances cannot be recovered.

After a notebook instance is deleted, the data stored in the mounted directory will be deleted, regardless of whether the notebook instance uses the default storage or an EVS disk for storage.

3.4 Changing a Notebook Instance Image

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

Constraints

The target notebook instance is stopped.

Procedure

- Log in to the ModelArts management console and choose **DevEnviron** > **Notebook** in the navigation pane on the left to switch to the notebook page.
- 2. In the notebook list, click **More** in the **Operation** column of the target notebook instance and select **Change Image**.

Figure 3-4 Change Image



3. In the **Change Image** dialog box, select a new image and click **OK**. After the modification, you can view the new image on the notebook list page.

3.5 Dynamically Expanding EVS Disk Capacity

Overview

If a notebook instance uses an EVS disk for storage, the disk is mounted to / home/ma-user/work/ of the notebook container and the disk capacity can be expanded by up to 200 GB when the instance is running.

Application Scenarios

During notebook development, select a small EVS disk capacity, for example, 5 GB, when creating a notebook instance because the storage requirements are low at the initial stage. After the development, a large volume of data must be trained. Then, expand the disk capacity to cost-effectively meet your service needs.

Restrictions

- The target notebook instance must use EVS for storage.
- Up to 100 GB can be expanded at a time. Additionally, the total capacity after expansion cannot exceed 4096 GB.
- If the original capacity of an EVS disk is 4096 GB, the disk capacity cannot be expanded.
- After the instance is stopped, the expanded capacity still takes effect.

Procedure

- Log in to the ModelArts management console. In the left navigation pane, choose DevEnviron > Notebook to switch to the new-version Notebook page.
- 2. Click the name of a running notebook instance. On the instance details page, click **Expansion**.

Figure 3-5 Instance details page

Name	notebook-da0c 🖉		
Status	ා Running(25 minutes left) ල		
ID	đ		
Storage Path	/home/ma-user/work/		
Storage Capacity	5 GB (EVS) Expansion		

3. Set the capacity to be expanded and click **OK**. **Expanding** shows that the capacity expansion is in progress. After the expansion, the displayed storage capacity is the expanded capacity.

Name	notebook-da0c 🖉	
Status	ම Running(23 minutes left) ල්	
ID	·• •	Ū
Storage Path	/home/ma-user/work/	
Storage Capacity	S GB (EVS) Expanding	

3.6 Changing the Flavor of a Notebook Instance

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

Figure 3-6 Expanding

Constraints

The target notebook instance is stopped.

Procedure

- Log in to the ModelArts management console and choose DevEnviron > Notebook in the navigation pane on the left to switch to the notebook page.
- 2. In the notebook list, click ^{*} in the **Flavor** column of the target notebook instance and choose the target flavor from the drop-down list.

Figure 3-7 Changing flavor

Create A maximum of 10 notebook instances can be created. You can create 6 more.				
Name ↓Ξ	Status ↓Ξ	Image	Flavor	Des
notebook-f4	Running(47 minutes	pytorch1.8-cuda10.2-cudnn7-ubuntu18.04	CPU: 2vCPUs 8GB 👻	1
notebook-e666	Stopped	pytorch1.8-cuda10.2-cudnn7-ubuntu18.04	CPU: 2vCPUs 8GB	4
notebook-f158	Stopped	pytorch1.4-cuda10.1-cudnn7-ubuntu18.04	GPU: 1*V100(32GB) CPU: 8vCPUs 64GF	8
notebook-783b	Stopped	pyspark2.4.5-ubuntu18.04	GPU: 2*V100(64GB) CPU: 16vCPUs 128	3GB

3.7 Modifying the SSH Configuration for Notebook

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

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

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

Constraints

The target notebook instance must be stopped.

Changing the Key Pair and the IP Addresses in the Whitelist

- Log in to the ModelArts management console and choose DevEnviron > Notebook in the navigation pane on the left to switch to the notebook page.
- 2. Click the target notebook instance. Enable remote SSH and change the key pair and whitelist.

NOTE

For manually enabled remote SSH, see **Figure 1**. After the SSH configuration is updated, the remote SSH function cannot be disabled.

For remote SSH enabled by default in the selected image, see Figure 2.

Update SSH Configuration				
★ Key Pair	▼ Create			
Whitelist				
	OK Canel			

Figure 3-9 Changing the whitelist and key pair

Figure 3-8 Update SSH Configuration

< notebook-90	< notebook-9c9d			
Name	noteboc 🖉		Flavor	CPU: 2vCPUs 8GB 👻
Status	Stopped		Image	tensorflow2.1-cuda10.1-cudnn7-ubuntu18.04
ID	f993	đ	Created At	Aug 05, 2022 09:03:57 GMT+08:00
Storage Path	/home/n		Updated At	Aug 05, 2022 09:07:00 GMT+08:00
Storage Capacity	50 GB (Default)			
Remote SSH			Address	
Whitelist	🖉		Authentication	KeyPair-1174 🖉

- Click

 and choose an existing key pair, or click Create to create a new key pair.
- For details about how to configure a whitelist, see Changing the Key
 Pair and the IP Addresses in the Whitelist. After you change the IP
 addresses, the existing links are still valid. After the links are released, the
 new links only from the changed IP addresses can be set up.

3.8 Viewing All Notebook Instances of an IAM Project

Any IAM user granted with the **listAllNotebooks** and **listUsers** permissions can click **View all** on the notebook page to view the instances of all users in the current IAM project.

NOTE

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

Assigning the listAllNotebooks Permission to an IAM User

- 1. Log in to the management console as a tenant user, hover the cursor over your username in the upper right corner, and choose **Identity and Access Management** from the drop-down list to switch to the IAM management console.
- On the IAM console, choose Permissions > Policies/Roles from the navigation pane, click Create Custom Policy in the upper right corner, and create two policies.

Policy 1: Create a policy that allows users to view all notebook instances of an IAM project, as shown in **Figure 1**.

- Policy Name: Enter a custom policy name, for example, Viewing all notebook instances.
- Policy View: Select Visual editor.
- Policy Content: Select Allow, ModelArts Service, modelarts:notebook:listAllNotebooks, and default resources.

Figure 3-10 Creating a custom policy

Policies/Roles / Create C	ustom Policy				
() You can use custo	m policies to supplement system-defined policies for fine-grained permissions management. Lean	n more			
* Policy Name 🚺	policy/ht3rw				
Policy View 2	Visual editor JSON				
* Policy Content	∧ 3 O Allow	Actions: 1	0 O M	C (Optional) Add request condition	Đ Ū
	Select al modelarts:notebook:list4iNotebooks			X Q	
	A 🗹 ListOnly				
	6 ☑ modelarts:notebook:listAlNotebooks Cuery the list of all development environment instances				
	Select Existing Policy/Role Add Permissions				
Description	Enter a brief description.				
		0/256			
Scope	Project-level services				
8	OK Cancel				

Policy 2: Create a policy that allows users to view all users of an IAM project.

- Policy Name: Enter a custom policy name, for example, Viewing all users of the current IAM project.
- **Policy View**: Select **Visual editor**.
- Policy Content: Select Allow, Identity and Access Management, iam:users:listUsers, and default resources.
- 3. In the navigation pane, choose **User Groups**. Then, click **Authorize** in the **Operation** column of the target user group. On the **Authorize User Group** page, select the custom policies created in **2**, and click **Next**. Then, select the scope and click **OK**.

After the configuration, all users in the user group have the permission to view all notebook instances created by users in the user group.

If no user group is available, create a user group, add users using the user group management function, and configure authorization. If the target user is not in a user group, you can add the user to a user group through the user group management function.

Enabling an IAM User to Start Other User's Notebook Instance

If an IAM user wants to access another IAM user's notebook instance through remote SSH, they need to update the SSH key pair to their own. Otherwise, error **ModelArts.6786** will be reported. For details about how to update a key pair, see **Modifying the SSH Configuration for Notebook**.

ModelArts.6789: Failed to find SSH key pair KeyPair-xxx on the ECS key pair page. Update the key pair and try again later.

4 JupyterLab

Operation Process in JupyterLab JupyterLab Overview and Common Operations JupyterLab Plug-ins Using ModelArts SDK Using the Git Plug-in

4.1 Operation Process in JupyterLab

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

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



1. Create a notebook instance.

On the ModelArts management console, create a notebook instance with a proper AI engine. For details, see **Creating a Notebook Instance**.

- 2. Use JupyterLab to access the notebook instance. For details, see Accessing JupyterLab.
- 3. Upload training data and code files to JupyterLab. For details, see Uploading Files from a Local Path to JupyterLab.
- 4. Compile and debug code in JupyterLab. For details, see JupyterLab Overview and Common Operations.

5. In JupyterLab, call the ModelArts SDK to create a training job for in-cloud training.

For details, see .

4.2 JupyterLab Overview and Common Operations

JupyterLab is the next-generation web-based interactive development environment of Jupyter Notebook, enabling you to compile notebooks, operate terminals, edit Markdown text, enable interaction, and view CSV files and images.

JupyterLab is the future mainstream development environment for developers. It has the same components as Jupyter Notebook, but offering more flexibility and powerful functions.

Accessing JupyterLab

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

- Log in to the ModelArts management console. Choose DevEnviron > Notebook in the navigation pane on the left. The notebook list of the new version is displayed.
- 2. Click **Open** in the **Operation** column of a running notebook instance to access JupyterLab.

Figure 4-2 Accessing a notebook instance

Create	A maximum	of 10 notebook instances can	be created. You can cr	eate 1 more.		C
Name ↓≡	Status ↓Ξ	Image	Flavor	Description	Created At ↓Ξ	Operation
notebook-5fa	Start	r * * * *	CPU: 2vCPUs 8GB -	🖉	2021/12/14 21:2	Open Start Stop
notebook-b8	Runni	pytc cu	CPU: 2vCPUs 8GB v.	🖉	2021/11/18 23:1	Open Start Stop

3. The **Launcher** page is automatically displayed. Perform required operations. For details, see **JupyterLab Documentation**.

Figure 4-3 JupyterLab homepage



D NOTE

The notebook and console kernels and versions displayed on the **Launcher** page vary depending on the AI engine based on which a notebook instance is created. Figure 2 shows an example only. Obtain the notebook and console kernels and versions on the management console.

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

Creating an IPYNB File in JupyterLab

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

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

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



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

Figure 4-5 Created IPYNB file



Creating a Notebook File and Accessing the Console

A console is a Python terminal, which is similar to the native IDE of Python, displaying the output after a statement is entered.

On the JupyterLab homepage, click a proper AI engine in the **Console** area to create a notebook file.

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

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



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

Figure 4-7 Creating a notebook file (console)



Editing a File in JupyterLab

JupyterLab allows you to open multiple notebook instances or files (such as HTML, TXT, and Markdown files) in one window and displays them on different tab pages.

In JupyterLab, you can customize the display of multiple files. In the file display area on the right, you can drag a file to adjust its position. Multiple files can be concurrently displayed.



Figure 4-8 Customized display of multiple files

When writing code in a notebook instance, you can create multiple views of a file to synchronously edit the file and view execution results in real time.

To open multiple views, open an IPYNB file and choose **File** > **New View for Notebook**.

Figure 4-9 Multiple views of a file



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

For example, install an external library Shapely.

pip install Shapely!

For example, obtain PythonPath.

lecho \$PYTHONPATH

Figure 4-10 Running code



Renewing or Automatically Stopping a Notebook Instance

If you enable auto stop when you created or started a notebook instance, the remaining duration for stopping the instance is displayed in the upper right corner of JupyterLab. You can click the time for renewal.

Figure 4-11 Remaining duration



Figure 4-12 Renewing an instance

Update a	Update auto-stop time			
Auto-stop time	Auto-stop time(hour) 1			
	UPDATE	CANCEL		

Common JupyterLab Buttons and Plug-ins

Figure 4-13 Common JupyterLab buttons and plug-ins

	+	Ð	±	G	\$*	
0	Filter files by	name			C	۲,
~	I /					
•>	Name			^	Last Modifi	ed
	• 🔲 Untitled.ipynb			seconds a	go	
\equiv						

 Table 4-1 JupyterLab buttons

Button	Description
+	Open the Launcher page, on which you can quickly create notebook instances, consoles, or other files.
	Create a folder.
±	Upload files.
C	Refresh the file directory.
♦\$ ⁺	Git plug-in, which can be used to access the GitHub code library associated with the notebook instance.

Table 4-2 JupyterLab plug-ins

Plug-in	Description
	List files. Click this button to show all files in the notebook instance.
0	Display the terminals and kernels that are running in the current instance.
•>	Git plug-in, which can be used to quickly access the GitHub code library.
° ¢	Property inspector.

Plug-in	Description
≣	Show the document organization.

Figure 4-14 Buttons in the navigation bar

A File Edit View Run Kernel Git Tabs Settings Help

Table 4-3 Buttons in the navigation bar

Button	Description
File	Actions related to files and directories, such as creating, closing, or saving notebooks.
Edit	Actions related to editing documents and other activities in the IPYNB file, such as undoing, redoing, or cutting cells.
View	Actions that alter the appearance of JupyterLab, such as showing the bar or expanding code.
Run	Actions for running code in different activities such as notebooks and code consoles.
Kernel	Actions for managing kernels, such as interrupting, restarting, or shutting down a kernel.
Git	Actions on the Git plug-in, which can be used to quickly access the GitHub code library.
Tabs	A list of the open documents and activities in the dock panel.
Settings	Common settings and an advanced settings editor.
Help	A list of JupyterLab and kernel help links.

Figure 4-15 Buttons in the menu bar of an IPYNB file

 Untitled1.ipynb 					
∎ + % ⊡ Ů ▶ ■ C	▶▶ Code	∽ (t) git		2 vCPU + 8 GiB	PyTorch-1.8 O
💻 Untitled.ipynb	×				
8 + % ⊡ ₿ ►	C	Markdown ~ 🕓	git	2 vCPU + 4 GiB	PyTorch-1.4 🔿 <

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

Button	Description
8	Save a file.

Button	Description
+	Add a new cell.
ж	Cut the selected cell.
Ū	Copy the selected cell.
Ċ	Paste the selected cell.
•	Execute the selected cell.
	Terminate a kernel.
C	Restart a kernel.
*	Restart a kernel and run all code of the current notebook again.
Code 🗸	There are four options in the drop-down list:
	Code (Python code), Markdown (Markdown code, typically used for comments), Raw (a conversion tool), and - (not modified)
U	View historical code versions.
git	Git plug-in. The gray button indicates that the plug-in is unavailable in the current region.
2 vCPU + 4 GiB	Instance flavor.
PyTorch-1.4	Kernel for you to select.
0	Code running status. indicates the code is being executed.

Monitoring Resources

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



Figure 4-16 Resource usage

4.3 JupyterLab Plug-ins

4.3.1 Code Parametrization Plug-in

The code parametrization plug-in simplifies notebook cases. You can quickly adjust parameters and train models based on notebook cases without complex code. This plug-in can be used to customize notebook cases for competitions and learning.

Use Guide

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

Figure 4-17 Viewing a code cell



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

ig	ure		0.5	noi	icui	Enu	01	coue	cens								
8	+	Ж		Ċ	►	C	**	Code	~	C	git						
	0								Add I Edit F	Form Form							•
									Cut C	Cells						Х	
									Сору	Cells						С	
									Paste	Cells	Below					V	
									Delet	e Cells	5					D, D	
									Split	Cell				(trl+Sh	nift+-	
									Merg	e Sele	cted Ce	ells			Shit	ft+M	
									Merg	e Cell	Above			Ctrl	Backs	pace	
									Merg	e Cell	Below			Ct	rl+Shit	ft+M	

Figure 4-18 Shortcut menu of code cells

Add Form

If you click **Add Form**, a code cell will be split into the code and form edit area. Click **Edit** on the right of the form to change the default title.

Figure 4-19 Two edit areas

#@title Default title text

Default title text

Edit Form

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

• You can set the form field type to **dropdown**, **input**, and **slider**. See **Figure 4**. Each time a field is added, the corresponding variable is added to the code and form areas. If a value in the form area is changed, the corresponding variable in the code area is also changed.

NOTE

When creating a dropdown form, click **ADD Item** and add at least two items. See **Figure 5**.

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

Default title text								
variable_name: 1 🗸	dropdown							
variable_name: ^{III} please input string here	input "							
variable_name: 😑 👘 0	slider							

Figure 4-21 Creating a dropdown form

Form field type	Variable type
dropdown	✓ string ✓
	Add Item
	option1 input a value
	option2 input a value
Variable name	
variable name	

Figure 4-22 Deleting a form

Default title text	+ ^ ¥ Ш
variable_name1: bb 🗸	⑪
variable_name2: [1, 2, 3]	⑪
variable_name3: {'1': 'a'; 'b':2}	
variable_name4: (1, 2, 3)	⑪

- If the form field type is set to **dropdown**, the supported variable types are **raw** and **string**.

Add new form field

Form field type		Variable typ	pe	
dropdown	~	string		~
		string		
		raw		
Variable name				
variable_name				
			Cancel	Save

- If the form field type is set to **input**, the supported variable types are **boolean**, **date**, **integer**, **number**, **raw**, and **string**.
- If the form field type is set to **slider**, the minimum value, maximum value, and step can be set.
- If you click **Hide code**, the code area will be hidden.

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

4.4 Using ModelArts SDK

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

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

Example Code

Create a tra	aining job.							
from modelarts	s.session import Session							
from modelarts.estimator import Estimator								
session = Session()								
estimator = Est	timator(
	modelarts_session=session,							
	framework_type='PyTorch',	# AI engine name						
	framework_version='PyTorch-1.0.0-python3.6',	# AI engine version						
	code_dir='/obs-bucket-name/src/',	# Training script directory						
	boot_file='/obs-bucket-name/src/pytorch_sentin	nent.py', # Training boot script						
directory								
-	log_url='/obs-bucket-name/log/',	# Training log directory						
	hyperparameters=[
	{"label":"classes",							
	"value": "10"},							
	{"label":"lr",							
	"value": "0.001"}							
],							
	output_path='/obs-bucket-name/output/',	# Training output directory						
	train_instance_type='modelarts.vm.gpu.p100',	# Training environment						
specifications								
	train_instance_count=1,	# Number of training nodes						
	job_description='pytorch-sentiment with Model	Arts SDK') # Training job description						
job_instance =	estimator.fit(inputs='/obs-bucket-name/data/tra	ain/', wait=False,						
job_name='my	_training_job')							
Obtain a m	adal list							
Obtain a m	iodet list.							
from modelart	s.session import Session							
from modelarts	s.model import Model							
session = Session	on()							
model_list_resp	<pre>p = Model.get_model_list(session, model_status=</pre>	"published", model_name="digit",						
order="desc")								
Obtain serv	vice details.							
from modelart	s session import Session							
from modelart	s model import Predictor							
session = Session	on()							

predictor_info_resp = predictor_instance.get_service_info()

4.5 Using the Git Plug-in

In JupyterLab, you can use the Git plug-in to clone the GitHub open-source code repository, quickly view and edit data, and submit the modified data.

Prerequisites

The notebook instance is running.

Starting the Git Plug-in of JupyterLab

In the notebook instance list, locate the target instance and click **Open** in the **Operation** column to go to the JupyterLab page.

Figure 1 shows the Git plug-in of JupyterLab.

Figure 4-23 Git plug-in



Cloning a GitHub Open-Source Code Repository

Alternatively, click the icon shown in the following figure to clone the GitHub open-source code repository.



Viewing a Code Repository

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



Figure 4-24 Opening the folder and starting the Git plug-in

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

Figure 4-25 Viewing a code repository

\bowtie	File Edit View	Run Kernel Git T	abs Settings	Help	
			۵ ۵	B C	🛛 Launcher
•	Current Reposite	ory nples			
0	Current Branch master			Ť	
	Change	es	History		
•	Frédéric Collonval	9c35013	3 weeks ago		
ß	working master ori Fix contentheader de	igin/HEAD origin/master		- 1	
°	Ahmed Fasih Example for MainAre	f05fb90 eaWidget's contentHead	3 weeks ago er (#185)	•	
	Jeremy Tuloup Merge pull request #	c0281bc ¢184 from krassowski/pa	3 months ago tch-1	•	
	Michał Krassowski Update documentati	fcc90b3 ion excerpt	3 months ago	•	
≔	Jeremy Tuloup Merge pull request #	dcb1fa6 ‡181 from jupyterlab/fix/	5 months ago update-3.1	•	
D NOTE

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

\bowtie	File	Edit	View	Run	Kernel	Git	Tabs	Settings	Help		
								æ	٢	C	🛛 Launcher
3	Ļ	Curren extens	t Reposi ion-exa	tory mples							
0	ų	Curren master	t Branch r	1						•	
•			Branch	nes				Tags			
•	Fi	lter							lew Brar	nch	
EQ.	ų	2.x									
B .	ų	master									
ę.,	ષ	origin/1	.х								
#	ų	origin/2	.х								
_	բ	origin/H	IEAD								
G	ր	origin/f	collonva	l/issue1	20						

Viewing Modifications

If a file in the code repository has been modified, you can view the modified file under **Changed** on the **Changes** tab page. Click **Diff this file** on the right of the file name to view the modifications.

Figure 4-26 Viewing modifications



Committing Modifications

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

8.6					Arian	8.00 min
1994	HIE EDIT VIEW KUN KERNEL GIT Tabs Settings Hel	2	The second se		Q利注約1	9.30 min
		0	is Launcher A C READMEING A			
0	extension-examples		HEAD	INDEX		
0	P Current Branch master	•	() 22 conda activate jupyterlab-extension-examples		() conda activate jupyterlab-extension-examples	
-	Changes History		54 # go to the helio world example	54 4	# go to the Louncher example of launcher	
٨	- Staged	(1)	56 57 # install the extension in editable made	56	<pre># install the extension in editable mode</pre>	
	♥ README.md	м	()		()	
6	+ Changed	(0)				
°4	+ Untracked	(0)				
Ť						
=						
_						
	-	_				
	Modify README.md					
	push to 51tHub demo					
	Commit					

Figure 4-27 Committing modifications

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

Figure 4-28 Checking whether the committing is successful

M	File Edit View Run Kernel Git Tabs Settings Help		⑦剩余时间:29 min
	0 😵 0	C Launchar X READMEmd X	
	Current Repository		Refresh
O	extension-examples	HEAD INDEX	
0	1/2 Current Branch	() 12 conda activate jupyterlab-extension-examples 13 conda activate jupyterlab-extension-examples	.ion-examples
-	Changes History	54 # go to the hello wonld example 54 # go to the launcher example 55 cd hello-world 55 cd launcher	
٠	marongjin b3384ec 24 seconds ago 💌 🌢	55 55	
	working master	5/ # install the extension in editable mode (i) 5/ # install the extension in editable mode (i)	ble mode
昏	Modify README.md		
	Frédéric Collonval 9c35013 3 weeks ago *		
°o	origin/HEAD origin/master		
	Fix contentheader dev dependencies (#186)		
	Ahmed Fasih 105fb90 3 weeks ago 🝷		
	Example for MainAreaWidget's contentHeader (#185)		

Click the **push** icon, which is equivalent to running the **git push** command, to push the code to the GitHub repository. After the pushing is successful, the message "Successfully completed" is displayed. If the token used for OAuth authentication has expired, a dialog box is displayed asking you to enter the user token or account information. Enter the information as prompted.

Figure 4-29 Pushing code to the GitHub repository

🛱 File Edit View Run Kernel Git Tabs Settings Help		③期余时间:28 min
0 0	☑ Launcher X D README.md X	
Current Repository	HEAD	Refresh
Current Branch master	52 conda activate jupyterlab-extension-examples	() 52 conda activate jupyterlab-extension-examples
Changes History	54 # go to the hello world example	54 # # go to the Launcher example
marongjin b3384ec 24 seconds ago motorsg master Modify README.md	57 # (install the extension in editable mode ()	$\frac{1}{10}$ = $\frac{1}{10}$ measurem $\frac{1}{10}$ = 0 install the extension in editable mode $\frac{1}{1000}$
Frédéric Collonval 9x35013 3 weeks ago inigin(HEAD) (origin(matter) Fix contentheader dev dependencies (#186)		
Ahmed Fasih 105fb90 3 weeks ago Example for MainAreaWidget's contentHeader (#185)		
Jeremy Tuloup c0281bc 3 months ago * Merge pull request #184 from krassowski/patch-1		
Michał Krassowski fcc90b3 3 months ago 👻 Update documentation excerpt		
Jeremy Tuloup dcb1fa6 5 months ago * Merge pull request #181 from jupyterlab/fix/update-3.1		
Frédéric Collonval 08067fc 6 months ago * Fix context-menu and documents		
Frédéric Collonval e5b003d 6 months ago - More fix		
Frédéric Collonval 58715ef 6 months ago 👻 Fix tests and linter		
Frédéric Collonval 7cd15ba 6 months ago 🔻 Fix linter		
Frédéric Collonval d650b01 6 months ago * Update the examples to use the new menu definitions from settings		
Carlos Herrero fe27afb 6 months ago * Adds a new example about documents using shared models (#163)		
Frédéric Collonval 9e43faa 6 months ago 👻 Upgrade playwright to 1.13.1 (#179)		
R Ely b41f2c1 6 months ago * Custom completer example (#169)		
Carlos Herrero 09d8de7 6 months ago •	Savino comoleted	Successfully pushed

After the preceding operations are complete, on the **History** tab page of the JupyterLab Git plug-in page, you can see that **origin/HEAD** and **origin/master** point to the latest push. In addition, you can find the corresponding information in the committing records of the GitHub repository.

5 Local IDE

Operation Process in a Local IDE Local IDE (PyCharm) Local IDE (VS Code) Configuring a Local IDE Accessed Using SSH

5.1 Operation Process in a Local IDE

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

1. Configure a local IDE.

Configure a local IDE on your PC.

2. Create a notebook instance.

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

- 3. Use the local IDE to remotely access ModelArts DevEnviron.
- 4. Upload data and code to the development environment.
 - Copy the code to the local IDE, which will automatically synchronize the code to the in-cloud development environment.
 - If the data is less than or equal to 500 MB, directly copy the data to the local IDE.
 - If the data is larger than 500 MB, upload it to OBS and then to the EVS disk.
- 5. Upload the training script and dataset to the OBS directory.
- 6. Submit a training job.

Perform this operation on the ModelArts management console.

- Submit a training job in the local IDE.

5.2 Local IDE (PyCharm)

5.2.1 Configuring a Local IDE Accessed Using PyCharm Toolkit

ModelArts provides the PyCharm plug-in PyCharm Toolkit for you to remotely access a notebook instance through SSH.

Prerequisites

PyCharm professional 2019.2 or later has been installed locally. Remote SSH applies only to the PyCharm professional edition.

Step 1 Download and Install PyCharm Toolkit

The PyCharm Toolkit package has been integrated into the ModelArts management console. Download and install PyCharm Toolkit.

For details, see **Downloading and Installing PyCharm Toolkit**.

Step 2 Log In to PyCharm Toolkit

and so PyCharm Toolkit can exchange data with ModelArts.

Obtain an access key (Creating Access Keys (AK and SK)) and use the key for login authentication (Using Access Keys for Login) so the local IDE can exchange data with the in-cloud environment.

Step 3 Create a Notebook Instance

Create a notebook instance with remote SSH enabled and whitelist configured. Ensure that the instance is running. For details, see **Creating a Notebook Instance**.

Step 4 Automatically Configure PyCharm Toolkit

 In the local PyCharm development environment, choose ModelArts > Connect To Remote > Remote Config and configure PyCharm Toolkit.

Figure 5-1	Remotely	connecting	to PyCharm	Toolkit
------------	----------	------------	------------	---------

<u>M</u> odelArts <u>H</u> elp pythonProject -	main.py
Edit Credential	
Connect To Remote 🔹 🕨	Remote Config
Edit Training Job Configuration	
Run Training Job	
Stop Training Job	

D NOTE

If **Connect To Remote** is unavailable, create a notebook instance with remote SSH enabled. For details, see **Creating a Notebook Instance**.

If the fault persists, check whether the PyCharm Toolkit version is the latest one. If not, download the latest version.

Before downloading PyCharm Toolkit, clear the browser cache. If PyCharm Toolkit of an earlier version has been downloaded, the browser cache may lead to the failure in downloading a new version.

2. All notebook instances with remote SSH enabled under the account are displayed. Choose the target instance from the drop-down list.

Figure 5-2 Notebook list

Notebook List	×
Notebook:	notebook-b7d1 🔹
RunningStatus:	RUNNING
Flavor:	modelarts.vm.cpu.2u
ImageName:	PyTorch1.4-CUDA10.1-cuDNN7-Ubuntu18.04
SshUrl:	ssh://ma-user@dev-modelarts
KeyPairName:	KeyPair-3x-bj4
KeyPair:	
PathMappings:	/home/ma-user/work/pythonProject
	Apply Cancel

- KeyPair: Select the locally stored key pair of the notebook instance for authentication. The key pair created during the notebook instance creation is saved in your browser's default downloads folder.
- PathMappings: Synchronization directory for the local IDE project and notebook, which defaults to /home/ma-user/work/Project name and is adjustable.
- 3. Click **Apply**. After the configuration is complete, restart the IDE for the configuration to take effect.

After the restart, it takes about 20 minutes to update the Python interpreter for the first time.

Step 5 Access a Notebook Instance Through PyCharm Toolkit

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

Figure 5-3 Starting the connection



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

Figure 5-4 Interrupting the connection



Step 6 Upload Local Files to the Notebook Instance

Code in a local file can be copied to the local IDE, which will automatically synchronize the code to the in-cloud development environment.

Initial synchronization

In the **Project** directory of the local IDE, right-click **Deployment** and choose **Upload to** *Notebook name* from the shortcut menu to upload the local project file to the specified notebook instance.

🔲 P	roject 🔻				€	ŧ	Ŧ	\$	—	樻 trai	n_mn
< 1	Recogn	i	New	۱.							fro
>	MA_	x	Cut	Ctrl+X							fro
~	train	「	Сору	Ctrl+C							ses
	>∎d		Copy Path								ttt
	ی روپ انگری	Ô	<u>P</u> aste	Ctrl+V							pri
			Find <u>U</u> sages	Alt+F7						6	pri
	👗 te		Find in Files	Ctrl+Shift+F							
	👸 tr		Repl <u>a</u> ce in Files	Ctrl+Shift+R							
> 11	I External		Inspect Code								
2	Scratche		<u>R</u> efactor	►							
			Clean Python Compiled Files								
			Add to F <u>a</u> vorites	Þ							
			<u>R</u> eformat Code	Ctrl+Alt+L							
			Optimi <u>z</u> e Imports	Ctrl+Alt+O							
			Open In	۲							
			Local <u>H</u> istory	►							
		G	Reload from Disk								
		→ ‡	Compare With	Ctrl+D							
			Mark Directory as								
		†ţ	Deployment	•	<u> </u>	load	to No	ote-e	Ebdž	Z	
			Remove BOM		<u>+</u> <u>D</u> o	wnlo	ad fro	om N	lote-	eEbdZ	
		<u>††</u>	Diagrams	►							
		0	Create Gist								
			Shutdown Kernel								
		Ť	ModelArts Upload								
		Ŧ	ModelArts Download	ł							

Figure 5-5 Synchronizing local data to a notebook instance

Follow-up synchronization

After modifying the code, press **Ctrl+S** to save it. The local IDE will automatically synchronize the modification to the specified notebook instance.

After PyCharm Toolkit is installed, **Automatic Upload** is automatically enabled in the local IDE for automatically uploading the files in the local directory to the target notebook instance. If **Automatic Upload** is not enabled, enable it by referring to the following figure.



Figure 5-6 Enabling Automatic Upload

Step 7 Remotely Debug the Code

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

Figure 5-7 Selecting a Python interpreter

Python Interpreter					
🙀 Note-eEbdZ Python 3.7.10 (sftp://ma-user@10.155.1a-user/anaconda3/envs/TensorFlow-2.1.0/bin/python)					
Interpreter Settings					
Add Interpreter					
🔋 Note-eEbdZ is disconnected Note-eEbdZ Python 3.7.10sorFlow-2.1.0/bin/python) 偹 476 of 1979M					

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

Figure 5-8 Runtime logs

<pre>Bright * C3 * C = & containing_service.py & & SEADME.md </pre> The Decise f Allociest The Decise f Allocies T	D	cTest) codes) 🐞 train_mnist_tf.py		
Ref Doctrist Op Image: Section of the section of t	g			💑 customize_service.py 🛛 🛔 train_mnist_tf.py 🖄 🏭 README.md 🗵
Run: 🖕 train mmist tf ×	🛻 🖸 Structure 📑 🗜 Proj	BOOCESE (ADOCTOS) Bit codes Bi		Q- Date W at the line of
		Run: 🔮 train mnist tf × ssh://ma-user@10.155.101.174:	30001/home/ma-	user/miniconda3/envs/Tensorflom-1.15.0/bin/python -u /home/ma-user/mork/testssh/testok/codes/train_mnist_tf.py

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

Figure 5-9 Setting runtime parameters (1)



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

Run/Debug Configurations				
+ - 喧日 ど 🔺 🖛 🗄	Name: Statistics			
Python				
Distriction of	Configuration Logs			
	Script path: 🔻	FlationSymmetry and Station (Station Systems)	Экинобринодер.ру	
> 🗲 Templates				
	Environment variables:	PYTHONUNBUFFERED=1		
	Python interpreter:	🍓 Project Default (Note-RKBYR Python	3.7.6 (sftp://ma-user@1866	Fæt38-171.05892/hor ▼
	Working directory:	Press Ages and a statistic statistic statistic statistics and the statistic statistic statistics and the statistics and the statistic statistics and the statisti	p	
	Path mappings:			
	☑ Add content roots to	PYTHONPATH		
	Add source roots to I	PYTHONPATH		
	🗌 Run with Python Cons			
	Redirect input from:			
	 Refore launch 			
	Select autori			
			ок	

Figure 5-10 Setting runtime parameters (2)

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

Figure 5-11 Running the program in debug mode

樻 custo	🚜 customize_service.py 🛛 🚜 train_mnist_tf.py 👋 🟭 README.md 🗵						
	TITUM LENSUTI LUW.CAMMPLES.LULUTIALS.MNIIST IMPOTT IMPUTT AUDIA						
	<pre>tf.flags.DEFINE_integer('max_steps', 10, 'number of training iterations.')</pre>						
	tf.flags.DEFINE_string('data_url', '/home/ma-user/work/ <u>testssh</u> /train/data/ <u>Mnis</u>						
	tf.flags.DEFINE_string('train_url', '/home/jnn/temp/delete', 'saved model dire						
	FLAGS = tf.flags.FLAGS						
	def main(*args):						
	# Train model						
	ss = 100						
	print(ss)						
	<pre>print('Training model')</pre>						

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



Figure 5-12 Viewing variable values in debug mode

5.2.2 Configuring a Local IDE Manually Accessed Using PyCharm

A local IDE supports PyCharm and VS Code. You can use PyCharm or VS Code to remotely connect the local IDE to the target notebook instance on ModelArts for running and debugging code.

This section describes how to use PyCharm to access a notebook instance.

Prerequisites

- PyCharm professional 2019.2 or later has been installed locally. The PyCharm professional edition is available because remote SSH applies only to the professional edition.
- A notebook instance has been created with remote SSH enabled. Ensure that the instance is running. For details, see **Creating a Notebook Instance**.
- The address and port number of the development environment are available. To obtain this information, go to the notebook instance details page.

Figure 5-13 Instance details page

Address	ssh://ma-user@dev	-modelarts-	huawe	icloud.com	30581
Authentication	KeyPair-e744	Add	lress	Port n	Imper

• The key pair is available.

A key pair is automatically downloaded after you create it. Securely store your key pair. If an existing key pair is lost, create a new one.

Step 1 Configure SSH

- 1. In your local PyCharm development environment, choose **File** > **Settings** > **Tools** > **SSH Configurations** and click **+** to add an SSH configuration.
 - Host: address for accessing the cloud development environment. Obtain the address on the page providing detailed information of the target notebook instance.
 - Port: port number for accessing the cloud development environment.
 Obtain the port number on the page providing detailed information of the target notebook instance.
 - **User name**: consistently set to **ma-user**.
 - Authentication type: key pair
 - Private key file: locally stored private key file of the cloud development environment. It is the key pair file automatically downloaded when you created the notebook instance.
- 2. Click local to rename the connection. Then, click **OK**.
- 3. After the configuration is complete, click **Test Connection** to test the connectivity.
- 4. Select **Yes**. If "Successfully connected" is displayed, the network is accessible. Then, click **OK**.
- 5. Click **OK** at the bottom to save the configuration.

Figure 5-14 Configuring SSH

The Day Ten Handron Store Brance, alls Toos, and Wannas Horizont Bab.			
DocTest Malabed			
¥ ■ Project + ② ÷ ♥ ー			¢ - s
			a × b
Image: A state of the state			
t modes 0 mabod		 motebook-samples-1610346734 most 	
di di as		 ipynb_checkpoints 	1 - 1 - 1
af assar		► 🖿 theia	â,
g, conspyton		 In abcd In associate? 	
& READWE md			200
C testmind.py			1
🛙 qi Ke osu funor (oʻta)		 Im images Im images 	
Illi External Libraries			
Scratches and Consoles	Colle City Chillen	► IIII model1	
		 Interconst data Interconsts 	
	Navination Bar Alt+Home	 Instruction 	
		ascend tf_sample.jpynb	
	Drop files here to open		
		Cifar10_me100.jpynb	
		El Inson resultation	
			•
		s pie.conf	
		💰 t10k-labels-idx1-ubyte.gz	
		train-labels-idx1-ubyte.gz	
& Run: testmind (1) >>			¢ -
<pre>np_resource = np.otype([("resource", np.uoyte, 1)]) ok</pre>	R 1426		ote host 10
Process finished with exit code 0	× 1437	Deployment configuration to 10.155.101.174 has been created.	ŧ,
III ≜ 1000 ▶ § Nun 17 File Transfer ♦ Python Console 12 Terminal Deployment configuration to 10.155.101.174 has been created. // Configure (34 misutes ago)	🗅 Updating skeletons.	Remote Python 3.64 (struensorflow-1.8	Event Log 8/bin/(python)

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

- 1. Choose **Tools** > **Start SSH Session** to access the cloud development environment.
- 2. Run the following command to view the Python virtual environments built in the current environment in the **README** file in **/home/ma-user/**: cat /home/ma-user/README

- 3. Run the **source** command to switch to a specific Python environment.
- 4. Run **which python** to obtain the Python path and copy it for configuring the Python interpreter on the cloud.

Figure 5-15 Obtaining the path to the virtual environment built in the development environment



Step 3 Configure a Python Interpreter

1. Choose **File** > **Settings** > **Project**: *Python project* > **Python Interpreter**. Then,

click was and **Add** to add an interpreter.

- 2. Select **Existing server configuration**, choose the SSH configuration from the drop-down list, and click **Next**.
- 3. Configure the Python interpreter.
 - Interpreter: Enter the Python path copied in step 1, for example, / home/ma-user/anaconda3/envs/Pytorch-1.0.0/bin/python.

If the path is **~/anaconda3/envs/Pytorch-1.0.0/bin/python**, replace **~** with **/home/ma-user**.

- Sync folders: Set this parameter to a directory in the cloud development environment for synchronizing local project directory files. A directory in / home/ma-user is recommended, for example, /home/ma-user/work/ projects, because other directories may be prohibited from accessing.
- 4. Click ! on the right and select **Automatically upload** so that the locally modified file can be automatically uploaded to the container.
- 5. Click Finish.

The local project file has been automatically uploaded to the cloud environment. Each time a local file is modified, the modification is automatically synchronized to the cloud environment.

In the lower right corner, the current interpreter is displayed as a remote interpreter.



Figure 5-16 Configuring a Python interpreter

Step 4 Install the Dependent Library for the Cloud Environment

After accessing the development environment, you can use different virtual environments, such as TensorFlow and PyTorch. However, in actual development, you need to install dependency packages. Then, you can access the environment through the terminal to perform operations.

Choose **Tools** > **Start SSH Session** and select the configured development environment. Run the **pip install** command to install the required dependency packages.



Step 5 Debug Code in the Development Environment

You have accessed the cloud development environment. Then, you can write, debug, and run the code in the local PyCharm. The code is actually executed in the cloud development environment, and the Ascend AI resources on the cloud are used. In this way, you compile and modify code locally and run the code in the cloud.

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

Figure 5-17 Debugging code



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

Figure 5-18 Setting runtime parameters



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

Figure 5-19 Code breakpoint



Figure 5-20 Debugging in debug mode



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

Figure 5-21 Debug mode

	<pre>16 17</pre>
Debuq: 🍦 train mnist tf 🛛	inain)
(같 Debugger Ⅰ3 Console	
Frames	
Image: MainThread ↑ ↓ Image: Transmitted topy:21 Image: Transmitted topy:251 Image: Transmitted topy:203 Image: Transmitted topy:203 Image: Transmitted top:200 Image: Transmitted	↓ ►]= args = [tuple: 1] [/home/ma-user/work/testssh/testok/codes/train_mnist_tf.py]

Before debugging code in debug mode, ensure that the local code is the same as the cloud code. If they are different, the line where a breakpoint is added locally may be different from the line of the cloud code, leading to errors.

When configuring a Python interpreter in the cloud development environment, you are advised to select **Automatically upload** so that any local file modification can be automatically uploaded to the cloud. If you do not select **Automatically upload**, manually upload the directory or code after you modify the local code. For details, see **Step 6 Upload Local Files to the Notebook Instance**.

5.3 Local IDE (VS Code)

5.3.1 Connecting to a Notebook Instance Through VS Code

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

• Connecting to a Notebook Instance Through VS Code with One Click (Recommended)

In this mode, click **Access VSCode** in the **Operation** column of a notebook instance on the ModelArts console to open VS Code and connect to the instance.

• Connecting to a Notebook Instance Through VS Code Toolkit (Recommended)

In this mode, log in to the ModelArts VS Code Toolkit plug-in and use it to connect to an instance.

• Manually Connecting to a Notebook Instance Through VS Code

In this mode, use the VS Code Remote-SSH plug-in to configure connection information and connect to an instance.

5.3.2 Installing VS Code

Download URL:

 URL for Windows: https://update.code.visualstudio.com/1.57.1/win32-x64user/stable

NOTE

Linux system users must install VS Code as a non-root user.

• URL for other OSs: https://code.visualstudio.com/updates/v1_57

Figure 5-22 VS Code download URL

May 2021 (version 1.57)	
Update 1.57.1: The update addresses these issues.	
The Workspace Trust feature addresses CVE-2021-34529.	Download VS Code based on your operating system.
Downloads: Windows: User System ARM Mac: Universal 64	bit Arm64 Linux: deb rpm tarball ARM snap

VS Code version requirements:

You are advised to use VS Code 1.57.1 or the latest version for remote connection.

```
Figure 5-23 VS Code installation guide in Linux
```

```
:~/VSCode$ sudo dpkg -i code_1.67.2-1652812855_amd64.deb
[sudo] password for dc:
(Reading database ... 200705 files and directories currently installed.)
Preparing to unpack code_1.67.2-1652812855_amd64.deb ...
Unpacking code (1.67.2-1652812855) over (1.67.2-1652812855) ...
Setting up code (1.67.2-1652812855) ...
Processing triggers for gnome-memus (3.13.3-11ubuntu1.1) ...
Processing triggers for desktop-file-utils (0.23-1ubuntu3.18.04.2) ...
Processing triggers for mime-support (3.60ubuntu1) ...
Processing triggers for shared-mime-info (1.9-2) ...
::~/VSCode$ code
```

5.3.3 Connecting to a Notebook Instance Through VS Code with One Click

Prerequisites

- The notebook instance with remote SSH enabled is running. For details, see **Creating a Notebook Instance**.
- You have downloaded the key file of the instance to a following local directory or its subdirectory based on your operating system:
 Windows: C:\Users\{{user}}
 Mac or Linux: Users/{{user}}

Procedure

- Step 1 Log in to the ModelArts management console. In the left navigation pane, choose DevEnviron > Notebook to switch to the new-version Notebook page.
- **Step 2** In the **Operation** column of a running instance, choose **More** > **Access VS Code**.

Figure 5-24 Accessing VS Code

Notebook Hot							
Warm Tip: The notebook instances in the 🧐	Running' state are being billed. I	Manually stop them if they are not in use.	If you enabled Auto Stop when creati	ing a notebook instance, the ins	ance will automatically stop running after t	he period you specified. Yo	u will not be billed after it has stop
Create A maximum of 1	0 notebook instances can be crea	ited. You can create 9 more.					View all
Search by job name by default.							
Name ↓Ξ	Status JΞ	Image	Flavor	Description	Created At JΞ	Created By	Operation
notebook	Running(59 minute	pytorch1.8-cuda10.2-cudnn7-ubuntu	CPU: 2vCPUs 8GB *		Dec 07, 2023 09:21:14 GMT+08:00		Open Start Stop More 🔺
							Delete
							Change Image
							Save Image
							Access VS Code

Step 3 If you have installed VS Code, click **Open**. The **Visual Studio Code** page is displayed.



Figure 5-25 Opening Visual Studio Code

If VS Code has not been installed, click **Windows** or **other OS** as required to download and install VS Code. For details about how to install VS Code, see **Installing VS Code**.

Figure 5-26 Downloading and Installing VS Code



Step 4 If the ModelArts VS Code plug-in has not been installed, click **Install and Open**. If you have installed the plug-in, perform **Step 5**.

Figure 5-27 Installing the VS Code plug-in



The installation takes about 1 to 2 minutes. After the installation is complete, a dialog box is displayed in the lower right corner. Then, click **Reload Window and Open**.

NOTE

This section uses VS Code 1.57.1 as an example. The **Reload Window and Open** dialog box may not be displayed when you install other versions of VS Code. In this case, perform **Step 5**.



Figure 5-28 Reload Window and Open

In the displayed dialog box, select **Don't ask again for this extension** and click **Open**.

×1 -	tile Edit Selection View Go Run Terminal Help	Ont Started - Visual Studio Code	∎∎∎∎ ⊂ ~ ~ →
Ð		×Q Get Started X	
	 Vou have not yet opened a folder. 		
	Open Folder		
	Opening a folder will close all currently open editors. To keep them open, add a tolder instead.		
	You can done a repository locally.	Visual Studio Code	
	Clone Repository	Editing evolved	
	To learn more about how to use git and source control in VS Code read our docs.	Start Walkthroughs	
		Visual Studio Code X	
		Allow an extension to open this URI? Discusser the best extension to make VS Code yours.	
		Modelhtt-HuwariCloud (huwaricloud-ai-modelants-woode-toolkir) wate to open a	
		www.kewi.kewi.kewi.kewi.kewi.kewi.kewi.k	
		Dont ask agán for this estension.	
		vscode extension UV, code patienticolly scade - bookst	
		modelarts-images-manager [33H: image-167] /data2/workspace/singsiacesiae/	
		workspace (55%) image 167/ /data2/workspace/singsiacoiss	
		✓ Show welcome page on startup	
	> TIMELINE		2 (

Step 5 Remotely connect to a notebook instance.

 Before the remote connection is executed, the system automatically searches for the key file. If the key is found, a new window will be displayed and the system connects to the instance. In this case, you do not need to select the key.



Figure 5-29 Remotely connecting to a notebook instance

- If the key file is not found, a dialog box is displayed. Select the correct key as prompted.
 - Figure 5-30 Selecting a key file \triangleright \square ... 🗙 Getting Started 🛛 🗙 Ωı ρ Getting Started مړ Get Started with VS Code 💡 Learn the Fundamentals 😫 Clone Git Repository.. L₀ Recent (i) Please select the key file, and due to security restrictions, . 6월 × please make sure the key [KeyPair-directory or its subdirectories. Select keypair... Navigate to ModelArts console £63 ⊗ 0 ∆ 0 ጽ 🕻 If an incorrect key is selected, a message will be displayed. Then, select the correct key as prompted.





When the information shown in the following figure is displayed, the instance is accessed.

Figure 5-32 Connection successful



The following error message indicates that accessing the instance failed. In this case, close the dialog box and view the output logs in the **OUTPUT** window. Then, check the **FAQs** and locate the cause.

Figure 5-33 Connection failed



----End

5.3.4 Connecting to a Notebook Instance Through VS Code Toolkit

This section describes how to use the ModelArts VS Code Toolkit plug-in to remotely connect to a notebook instance.

Prerequisites

You have downloaded and installed VS Code. For details, see Installing VS Code.

Step 1 Install the VS Code Plug-in

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

Figure 5-34 Install the VS Code Plug-in



2. Wait for about 1 to 2 minutes.

Figure 5-35 Installation process



3. After the installation is complete, check the message displayed in the lower

right corner. If the ModelArts icon and remote SSH icon are displayed in the navigation pane on the left, the VS Code plug-in is installed.

Figure 5-36 Installation completion message

Completed installing ModelArts extension from VSIX.



Figure 5-37 Installation completed

Network issues may cause an installation failure. If this occurs, proceed with follow-up operations. After 1 in **Step 5 Connect to the Notebook Instance** is performed, the system will automatically display a dialog box shown in the following figure. In this case, click **Install and Reload**.

Figure 5-38 Reconnecting remote SSH

8		R ch-	i Extension 'Remote - SSH' is required to open the r Do you want to install the extension?	remote window. X
£073		Configu		Install and Reload
▲ Disc	connected from vscode-remote	⊗ 0 ∆ 0		R 🗘

Step2 Adding More Regions

1. Click Get More Region.

ModelArts Login	
Account Name	
AccessKey Id	Get Your AK/SK
SecretAccess Key	
	Get More Region
Log in	

- 2. Go to the folder corresponding to your region. (View the region information in the console address.)
- 3. Click the YAML file, right-click **Raw**, and choose **Copy link address** from the shortcut menu to copy the file address.

Obtain the host information.

The plug-in must be used by calling APIs. The API domain names of some HCSO regions are not registered. You need to configure the corresponding IP addresses and API domain names in the **hosts** file on the local PC. The **hosts** file is generally stored in **C:\Windows\System32\drivers\etc**.

4. Import the configuration file in the VS Code plug-in.

Open the VS Code plug-in. Click , choose **Import Region Profile**, click **From url** in the lower right corner, enter the URL of the YAML configuration file, and press **Enter**.

5. Log in to the VS Code plug-in to use more functions.

After the configuration file is imported, the region changes to your region. Enter the account name and AK/SK to log in to the plug-in.

Note: If the configuration file cannot be downloaded from the URL copied in Gitee, perform the following steps:

a. Right-click **Raw** and choose **Save link as** from the shortcut menu to save the file to the local PC.

b. Open the VS Code plug-in. Click and choose **Import Region Profile**. In the dialog box that is displayed in the lower right corner, click **From local file** and select the downloaded **ModelArts-region-profile.yaml** file.

Step 3 Log In to the VS Code Plug-in

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

Figure 5-39 Logging in to the plug-in

ModelArts Login	
Name	
AK	Get Your AK/SK
SK	
	~
Log in	
JSON Log	in

Enter the login information and click **Log In**.

- Name: Custom username, which is displayed only on the VS Code page and is not associated with any account.
- AK and SK: Access key pair. To create a key pair, log in to Huawei Cloud, choose My Credentials > API Credentials > Access Keys, and click Create Access Key.
- Region: must be the same as that of the notebook instance to be remotely connected. Otherwise, the connection will fail.

Alternatively, you can switch the login mode, enter your login information, and press **Ctrl+S** to save the information.

Figure 5-40 Configuring login information

F	user-in	fo	×			
C:	> User	s > 🕅	兴 率	************************************	arts.modelarts -0.826 > .profile >	≣ user-info
		[
	2		{			
				"name": "",		
				"ak": "",		
				"sk": "",		
	6			"region": "		
			}			
		1				

2. After the login, check the notebook instance list.

Figure 5-41 Login succeeded

>	<u>F</u> ile	<u>E</u> dit	Selection	<u>V</u> iew	<u>G</u> o	<u>R</u> un	<u>T</u> erminal
Ch		MODEL	ARTS			വ	© ©
	~	<u>8</u>	19.2013-2013	<i>.</i> 0320	13. A	2	
Q		۱	notebook-8	868c (RL	JNNIN	lG)	
		۹	notebook-l	99e8 (ST	OPPE	D)	
2º							
8							

Step 4 Create a Notebook Instance

• Create a notebook instance with remote SSH enabled, and download the key file to either of the following directories based on your OS:

Windows: C:\Users\{{user}}

Mac or Linux: Users/{{user}}

• A key pair is automatically downloaded after you create it. Securely store your key pair. If an existing key pair is lost, create a new one.

Create a notebook instance with remote SSH enabled. For details, see **Creating a Notebook Instance**.

Step 5 Connect to the Notebook Instance

1. In the local VS Code development environment, right-click the instance name and choose **Connect to Instance** from the shortcut menu to start and connect to the notebook instance.

The notebook instance can either be running or stopped. If it is stopped, the VS Code plug-in starts the instance and then connects to it.



Figure 5-42 Connecting to a notebook instance

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





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

Figure 5-44 Configuring the key file



3. Wait for about 1 to 2 minutes until the notebook instance is accessed. After information similar to the following is displayed in the lower left corner of the VS Code environment, the connection is succeeded.

Figure 5-45 Connection succeeded



Related Operations

For details about uninstalling the VS Code plug-in, see Figure 13.





Figure 5-46 Step 1 Download the VS Code Plug-in

5.3.5 Manually Connecting to a Notebook Instance Through VS Code

A local IDE supports PyCharm and VS Code. You can use PyCharm or VS Code to remotely connect the local IDE to the target notebook instance on ModelArts for running and debugging code.

This section describes how to use VS Code to access a notebook instance.

Prerequisites

- You have downloaded and installed VS Code. For details, see Installing VS Code.
- Python has been installed on your local PC or server. For details, see VS Code official documentation.

- A notebook instance has been created with remote SSH enabled. Ensure that the instance is running. For details, see **Creating a Notebook Instance**.
- The address and port number of the development environment are available. To obtain the information, go to the notebook instance details page.

Figure 5-47 Instance details page

Address	ssh://ma-user@dev-	modelarts-	80581
Authentication	KeyPair-e744	Address	Port number

• The key pair is available.

A key pair is automatically downloaded after you create it. Securely store your key pair. If an existing key pair is lost, create a new one.

Step 1 Add the Remote-SSH Plug-in

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



Step 2 Configure SSH

1. In the local VS Code development environment, click 40 on the left, select SSH Targets from the drop-down list box, and click 📖. The SSH configuration file path is displayed.



Figure 5-49 Configuring SSH Targets

2. Click the SSH configuration path and configure SSH.

Figure 5-50 SSH configuration file path

Select SSTECOTINGUIATION THE TO UPDATE

C:\Users\/22224\.ssh\config

C:\ProgramData\ssh\ssh_config

Settings specify a custom configuration file

Help about SSH configuration files

HOST remote-dev

hostname <*instance connection host>* port <*instance connection port>* user **ma-user** IdentityFile **~/.ssh/test.pem** UserKnownHostsFile=/dev/null StrictHostKeyChecking no

- **HOST**: name of the cloud development environment
- HostName: address for accessing the cloud development environment. Obtain the address on the page providing detailed information of the target notebook instance.
- Port: port number for accessing the cloud development environment.
 Obtain the port number on the page providing detailed information of the target notebook instance.
- user: ma-user
- IdentityFile: locally stored private key file of the cloud development environment. It is the key pair file in Prerequisites.
- Choose File > Preference > Settings > Extensions > Remote-SSH. On the Remote Platform page, click Add Item, set Item and Value, and click OK.

HTML	common install locations.			
Jake				
JavaScript Debugger				
JSON				
Jupyter 😂	Remote.SSH: Remote I	Platform		
LESS	A map of the remote hostname to the platform for that remote. Valid values:			
Markdown	linux, macOS, windows	. Note - this setting will led so it is currently bei	ing autopopulated f	vnen for successful
Merge Conflict	connections, but is not currently used.			
Node debug	la sur	Value		
Npm	item	value		
РНР	remote-dev	linux		
Python	test	linux	~ ОК	Cancel
Reference Search V				
Remote - Containers				
Remote - SSH	Remote.SSH: Remote S	Server Listen On Socket		

Figure 5-51 Configuring Remote Platform

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

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


Figure 5-52 Opening the development environment

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







Figure 5-54 Complete configuration example

Step 3 Install the Python Plug-in in the Cloud Development Environment



On the displayed VS Code page, click on the left, enter **Python** in the search box, and click Install.

Figure 5-55 Installing the Python plug-in in the cloud development environment



If the Python plug-in fails to be installed on the cloud, install it using an offline package.

Step 4 Install the Dependent Library for the Cloud Environment

After accessing the container environment, you can use different virtual environments, such as TensorFlow and PyTorch. However, in actual development, you need to install dependency packages. Then, you can access the environment through the terminal to perform operations.

In VS Code, press Ctrl+Shift+P. 1.

- 2. Search for **Python: Select Interpreter** and select the target Python.
- 3. Choose **Terminal > New Terminal**. The CLI of the remote container is displayed.
- 4. Run the following command to install the dependency package: pip install spacy

5.3.6 Remotely Debugging in VS Code

Prerequisites

A notebook instance has been accessed through VS Code.

Step 1 Upload Local Code to the Cloud Development Environment

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

Figure 5-56 Open Folder

Дh	EXPLORER ····	
E,	✓ NO FOLDER OPENED	
Q	Connected to remote.	
gо	Open Folder	
å	You can also clone a repository from a URL. To learn more about how to use git and source control in VS Code read our docs.	
Ŀø	Clone Repository	

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

Figure 5-57 Selecting a file path

Open Folder		
/home/ma-user/	ОК	Show Local
*		
.astropy		
.cache		
.conda		
.config		
.jupyter		
.local		
.modelarts		

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

Step 2 Debug Code Remotely

Open the code file to be debugged in VS Code. Before running the code, click the default Python version in the lower left part and select a version as required.

 MA-USER [SSH: REMOTE-DI . ssh . vscode . vscode . vscode . vscode-server . vscode-server . vscoda . vscode-server . vs	
 Assh Assh Asscode Asscode Asscode Asscode-server Astreter Astreter	
> .vscode Python 2.7.12 64-bit > .vscode-server /usr/bin/python > .yarn Python 3.5.2 64-bit > anaconda3 /usr/bin/python3 > env_script Python 3.6.12 64-bit (conda) > log ./anaconda3/envs/R-3.6.1/bin/python > modelarts-sdk Python 3.6.2 64-bit (conda) > notebook-exts ./anaconda3/envs/PySpark-2.3.2/bin/python > notebook-samples Python 3.6.2 64-bit (conda) > work .bash_logout > bash_logout .bashrc @ npmrc .profile	
 > .vscode-server .yarn .yarn Python 3.5.2 64-bit anaconda3 /usr/bin/python3 env_script Python 3.6.12 64-bit (conda) log ./anaconda3/envs/R-3.6.1/bin/python modelarts-sdk Python 3.6.2 64-bit (conda) . notebook-exts ./anaconda3/envs/PySpark-2.3.2/bin/python notebook-samples Python 3.6.2 64-bit (conda) . notebook-samples Python 3.6.2 64-bit (conda) .anaconda3/envs/XGBoost-Sklearn/bin/python 	
 > .yarn Python 3.5.2 64-bit > anaconda3 //usr/bin/python3 > env_script Python 3.6.12 64-bit (conda) > log ,/anaconda3/envs/R-3.6.1/bin/python > modelarts-sdk Python 3.6.2 64-bit (conda) > notebook-exts ,/anaconda3/envs/PySpark-2.3.2/bin/python > notebook-samples Python 3.6.2 64-bit (conda) > notebook-samples Ibid //anaconda3/envs/XGBoost-Sklearn/bin/python > test_456064 > PythonPractise @ README.md * test.py > work Bash_logout Bashric PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 2: Pyth (base) sh-4.3\$/home/ma-user/anaconda3/envs/TensorFlow-1.8 	
 > anaconda3 > env_script > log > hodelarts-sdk > python 3.6.12 64-bit (conda) > notebook-exts > notebook-exts > notebook-samples > python 3.6.2 64-bit (conda) > anaconda3/envs/XGBoost-Sklearn/bin/python > test_456064 > PythonPractise > bash_logout > bash_logout > bashric > profile PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 2: Pyth (base) sh-4.3\$/home/ma-user/anaconda3/envs/TensorFlow-1.8 	
 > env_script > log > log > modelarts-sdk > python 3.6.12 64-bit (conda) > notebook-exts > notebook-exts > notebook-samples > python 3.6.2 64-bit (conda) > anaconda3/envs/XGBoost-Sklearn/bin/python > test_456064 > PythonPractise > bash_logout > bash_logout > bashrc > profile PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 2: Pyth (base) sh-4.3\$/home/ma-user/anaconda3/envs/TensorFlow-1.8 	
 > log	
 modelarts-sdk Python 3.6.2 64-bit (conda) /anaconda3/envs/PySpark-2.3.2/bin/python notebook-samples Python 3.6.2 64-bit (conda) notebook-samples Python 3.6.2 64-bit (conda) notebook-samples-16(/anaconda3/envs/XGBoost-Sklearn/bin/python test_456064 PythonPractise README.md test.py work .bash_logout .bash_logout .bashr ProBLEMS OUTPUT DEBUG CONSOLE TERMINAL 2: Pyth (base) sh-4.3\$/home/ma-user/anaconda3/envs/TensorFlow-1.8 	
 notebook-exts /anaconda3/envs/PySpark-2.3.2/bin/python notebook-samples Python 3.6.2 64-bit (conda) notebook-samples-16(/anaconda3/envs/XGBoost-Sklearn/bin/python test_456064 PythonPractise README.md test.py work .bash_logout .bash_logout .bashrc PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 2: Pyth (base) sh-4.3\$/home/ma-user/anaconda3/envs/TensorFlow-1.8 	
 > notebook-samples > notebook-samples-16(/anaconda3/envs/XGBoost-Sklearn/bin/python > test_456064 > PythonPractise @ README.md * test.py > work = .bash_history = .bash_logout = .bashrc PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 2: Pyth (base) sh-4.3\$/home/ma-user/anaconda3/envs/TensorFlow-1.8, 	
 > notebook-samples-16(/anaconda3/envs/XGBoost-Sklearn/bin/python > test_456064 > PythonPractise @ README.md * test.py > work = .bash_history = .bash_logout = .bashrc PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 2: Pyth (base) sh-4.3\$/home/ma-user/anaconda3/envs/TensorFlow-1.8; 	
v test_456064 > PythonPractise (i) README.md I test.py > work I .bash_history I .bash_logout I .bashrc PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 2: Pyth (base) sh-4.3\$/home/ma-user/anaconda3/envs/TensorFlow-1.8	
> PythonPractise © README.md • test.py > work E .bash_history .bash_logout .bashrc PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 2: Pyth (base) sh-4.3\$/home/ma-user/anaconda3/envs/TensorFlow-1.8;	
● README.md ● test.py > work ■ .bash_history ■ .bash_logout ■ .bashrc PROBLEMS OUTPUT □.profile (base) sh-4.3\$/home/ma-user/anaconda3/envs/TensorFlow-1.8,	
> work .bash_history .bash_logout .bashr.c .pmrc .profile (base) sh-4.3\$/home/ma-user/anaconda3/envs/TensorFlow-1.8, PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 2: Pyth	
F.bash_history I.bash_logout I.bashrc I.bashrc I.profile (base) sh-4.3\$/home/ma-user/anaconda3/envs/TensorFlow-1.8;	
Image: starting s	
image: sportile image: sportile PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 2: Pyth image: sportile (base) sh-4.3\$/home/ma-user/anaconda3/envs/TensorFlow-1.8;	
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 2: Pyth profile (base) sh-4.3\$/home/ma-user/anaconda3/envs/TensorFlow-1.8;	
profile (base) sh-4.3\$/home/ma-user/anaconda3/envs/TensorFlow-1.8	on 🗸 🚽
(base) sn-4.3\$/nome/ma-user/anaconda3/envs/lensorFiow-1.8	1. to to the second
≣ .python_history st nv	bin/python /home/ma-
≣ viminfo 1	
⊊ .varnrc 2	
3 ≣ core,9193	
(base) sn-4.33	
Q ≣ core.9751	
E core.9929	
A ≡ core 10103	
> OUTLINE	
SSH: remote-dev Python 3.6.4 64-bit (conda) 🛞 0 🛆 0 👷 No Ports Available 🤣 Ln 4.	

Figure 5-58 Selecting a Python version

- Click the execution button to run the code. The code output is shown on the **TERMINAL** tab page.
- If a training job takes a long time to execute, run the job at the backend through the nohup command. This prevents the disconnection of an SSH session or a network failure from affecting job execution. The following shows an example nohup command: nohup your_train_job.sh > output.log 2>&1 & tail -f output.log
- To debug the code, perform the following operations:
 - a. Choose **Run > Run and Debug** on the left.
 - b. Select the default Python code file.
 - c. Click on the left of the code to set breakpoints.
 - d. Debug the code according to the debug procedure which is displayed above the code, and the debug information is displayed on the left of the page.

5.3.7 Uploading and Downloading a File in VS Code

Uploading Data from a Local IDE to a Notebook Instance

If the data is less than or equal to 500 MB, directly copy the data to the local IDE.

If the data is larger than 500 MB, upload the code to OBS and then to the EVS disk associated with the target notebook instance.



Figure 5-59 Uploading data to a notebook instance through OBS

Procedure

Upload data to OBS. . Alternatively, use ModelArts SDK on a local VS Code terminal.

The following shows how to enable Terminal in the VS Code environment.



Figure 5-60 Enabling Terminal in the local VS Code environment

The following shows how to use ModelArts SDK on a local VS Code terminal to upload a local file to OBS:

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

Then, upload the file to OBS by referring to Uploading a File to OBS.

				5				
∢							Get Started - Visual Studio Co 🔳 🔲 🚺 🛙	
Ð	×	Get S	Started $ imes$					
þ			Start				Walkthroughs	

Figure 5-61 Enabling Terminal in the remote VS Code environment



2. The following shows how to use ModelArts SDK in the terminal of the remote VS Code environment to download files from OBS to a development environment:

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

Then, upload the file to OBS by referring to Uploading a File to OBS.

Downloading Files from a Notebook Instance to a Local Directory

Files created in Notebook can be downloaded to a local path. The operations for downloading a file are the same, regardless of whether the created notebook instance uses the default or EVS storage. In the **Project** directory of the local IDE, right-click the **Notebook2.0** project and choose **Download** from the shortcut menu to download the project file to the local PC.

Figure 5-62 Downloading files from a notebook instance to a local directory in VS Code

∢	<u>F</u> ile <u>E</u> dit <u>S</u> e	election <u>V</u> i	ew G	io <u>R</u> un	<u>T</u> ern	minal	<u>H</u> elp	Getting Sta	arted -	- work [SS	SH: Model	Arts-Not	te-Y30	CXa] - V	isual Stu	udio Code			×
വ	EXPLORER					刘 G	etting Star	ted ×											
		ORS																	
0	🗙 🐋 Gei	tting Started																	
	V WORK [SSH	: MODELART	S-NOTE	-Y3CXA]															
0	> .ipynb_c	heckpoints				St	art					Ge	ttin	g Sta	rted				
ج	> .ma_proj	ject				C.	New File												
	> asset					ም ዓ	Onen Fil					_	Ş	Get St	arted v	vith VS (Code		
	E jupyterla	ab-filebrows	er-2.0.	6.tgz		R	Onen Fo	lder											
	📑 Untitle ^{, *}						openito						•	Learn	the Fu	ndamen	tals		
	Untitle	Open to	the Sid	de		(trl+Enter.												
0		Open W	ith										ଚ	Boost	your P	roductiv	vity		
		Open in	Integra	ated Terr	minal														
E 4		Coloct fo	Com																
			or Com	pare				1 D:\C	ode\\	vscode_te	oolkits\op	oe							
1284		Cut					Ctrl+X		D:\	\Code\vs	code_too	lk							
		Copy					Ctrl+C	0060447	75										
								Jsers											
		Downloa	ad						D:\	\Code\vs	code_too	olk							
		Сору Ра	th			Sł	ift+Alt+C												
		Copy Re	lative F	Path	Ct	rl+K Ctr	l+Shift+C												
8																			
		Réname					F2			🗹 S	Show welc	come pa	ige o	n startı	qu				
563		Delete F	erman	ently			Delete						Page						
~1~	> OUTLINE																		
× s:	SH: ModelArts-No	ote-Y3CXa	⊗ 0 ∆	70 🕅	0													<i>R</i>	¢

5.4 Configuring a Local IDE Accessed Using SSH

This section describes how to use PuTTY to remotely log in to a notebook instance on the cloud in the Windows environment.

Prerequisites

- You have created a notebook instance with remote SSH enabled and whitelist configured. Ensure that the instance is running. For details, see **Creating a Notebook Instance**.
- The address and port number of the development environment are available. To obtain this information, go to the notebook instance details page.

Figure 5-63 Instance details page



• The key pair is available.

A key pair is automatically downloaded after you create it. Securely store your key pair. If an existing key pair is lost, create a new one.

Step 1 Install the SSH Tool

Download and install the SSH remote connection tool, for example, PuTTY.

Step 2 Use PuTTYgen to Convert the .pem Key Pair File to a .ppk Key Pair File

- 1. Download PuTTYgen and double-click it to run it.
- 2. Click **Load** to load the .pem key file created and saved during notebook instance creation.
- 3. Click **Save private key** to save the generated .ppk file. The file name can be customized, for example, **key.ppk**.

Figure 5-64 Converting the .pem key pair file to a .ppk key pair file

e Key Convers	ions Help				
Key					
Public key for pasting	into OpenSSH a	uthorized_keys file:			
ssh-rsa IzaliAM2tin-M2tin-975	たいためわずいためわずいたい	NG BARATAN BARANG	NASER KSELANASER AND	BERKANA KARAMATAN	1
สพิสส์พลิพิสีพิสีส	<u> </u>	antitican Seria fresiant			
HER DESCRIPTION OF THE	are deste anste Krief Grief des	io più io sel anno 2 apr Toma toma toma toma toma toma toma toma t	elan Mercindercindert XI		
Key fingemrint:	sshirsa 2048 SH	A256 Partice and the Transf	ເວລາວ ແລະ ເອກະ ແລະ ເ	12. The State of t	_
Key ningerprint.			e nationalese me unhant me omsee mationales h	n naise sen dengel o	
Key comment:	imported-openss	h-key			
Key passphrase:					
Confirm passphrase:					
Actions					
Generate a public/priv	vate key pair			Generate	
Load an existing priva	te key file			Load	
Save the generated k	ey		Save public key	Save private key	
Parameters					
Type of key to genera	te: ODSA	○ ECDSA	OEdDSA	O SSH-1 (RSA)	
U NJA	<u> </u>			_	

Step 3 Use SSH to Connect to a Notebook Instance

1. Run PuTTY.

- 2. Click **Session** and set the following parameters:
 - a. **Host Name (or IP address)**: address for accessing the in-cloud notebook instance. Obtain the address on the page providing detailed information of the target notebook instance .
 - b. **Port**: port number for accessing the in-cloud notebook instance. Obtain the port number on the page providing detailed information of the target notebook instance, for example, **32701**.
 - c. Connection type: SSH
 - d. **Saved Sessions**: task name, which can be clicked for remote connection when you use PuTTY next time

Reputer Configuration ? × Category: Session Basic options for your PuTTY session ···· Logging Specify the destination you want to connect to — Terminal а Host Name (or IP address) D Port -- Keyboard dev-modelarts-cnnorth4.huaweicloud.cor 32701 - Bell --- Features Connection type: С . ⊡ · Window O Serial O Other: SSH Telnet \sim Appearance Behaviour Load, save or delete a stored session Translation Saved Sessions Selection diama dia mangana dia manga Mangana dia mang Colours - Connection Default Settings Load **1** ···· Data Save Proxy Delete Serial Telnet - Rlogin --- SUPDUP Close window on exit: ○ Always ○ Never Only on clean exit About Help Open Cancel

Figure 5-65 Configuring Session

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

🕵 PuTTY Configuration	? ×
Category:	
Session Logging Generation Connection Ferial France Selection Connection Proxy SSH Serial Telnet Rlogin SUPDUP	Options controlling character set translation Character set translation Remote character set: UTF-8 (Codepages supported by Windows but not listed here, such as CP866 on many systems, can be entered manually) Treat CJK ambiguous characters as wide Caps Lock acts as Cyrillic switch Adjust how PuTTY handles line drawing characters Handling of line drawing characters: Use Unicode line drawing code points Poor man's line drawing (+, - and I) Font has XWindows encoding Use font in both ANSI and OEM modes Use font in OEM mode only Copy and paste line drawing characters as lqqqk Enable VT100 line drawing even in UTF-8 mode
About Help	Open Cancel

Figure 5-66 Setting the character format

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

🕵 PuTTY Configuration			?	×				
Category:								
	Data to send to the server							
Terminal	Login details							
Keyboard	Auto-login username	ma-user						
Bell Features	When usemame is not specified: Prompt Ouse system usemame (100443458)							
⊡ Window Appearance	Terminal details							
Behaviour	Terminal-type string	xterm						
	Terminal speeds	38400,384	00					
Colours	Environment variables							
Data	Variable		A	dd				
···· Proxy ⊕·· SSH	Value		Ren	nove				
Serial Telnet Rlogin								
SUPDUP								
About Help		Open	Cano	el				

Figure 5-67 Entering a username

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



6. Click **Open**. If you are logging in to the instance for the first time, PuTTY displays a security warning dialog box, asking if you want to accept the instance security certificate. Click **Accept** to save the certificate to your local registry.

Figure 5-68 Asking if you want to accept the instance security certificate



7. Connect to the notebook instance.

Figure 5-69 Connecting to a notebook instance

🗬 dev-modelarts-cnnorth4.huaweicloud.com - PuTTY	-		Х
³ Using username "ma-user". ³ Authenticating with public key "imported-openssh-key" ³ Authenticating with public (GNU/Linux 3.10.0-862.14.1.5.h328.6 ⁴ x86_64)	euleros	v2r7.3	<86_
<pre>* Documentation: https://help.ubuntu.com * Management: https://landscape.canonical.com * Support: https://ubuntu.com/advantage</pre>			
This system has been minimized by removing packages and content that required on a system that users do not log into.	that ar		
fo restore this content, you can run the 'unminimize' command. Last login: Thu Aug 12 15:10:57 2021 from 2012 (2013) sh-4.4\$			

6 ModelArts Tool Guide

PyCharm Toolkit Preparations PyCharm Toolkit (Latest Version) FAQs

6.1 PyCharm Toolkit

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

Constraints

- Currently, only PyCharm 2019.2 or later is supported, including the community and professional editions.
- Only PyCharm of the professional edition can be used to access the notebook development environment.
- You can use a community or professional edition of PyCharm Toolkit to submit training jobs. PyCharm Toolkit 2.x can be used only to submit training jobs of the old version, and the latest version of PyCharm Toolkit can be used only to submit training jobs of the new version.
- PyCharm Toolkit supports PyCharm of the Windows, Linux, or Mac version.

Available Functions

Function	Description	Reference
Remote SSH	The notebook development environment can be accessed through remote SSH.	Configuring PyCharm Toolkit to Remotely Connect to a Notebook Instance
Model training	Code developed locally can be quickly submitted to ModelArts and a training job of the new version is automatically created. During the running of the training job, training logs can be obtained and displayed on a local host.	 Submitting a Training Job (New Version) Stopping a Training Job Viewing Training Logs

Table 6-1 Toolkit functions of the latest version

6.2 Preparations

6.2.1 Downloading and Installing PyCharm Toolkit

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

Prerequisites

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

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

Method 1: Install PyCharm Toolkit in Marketplace

Choose File > Settings > Plugins, search for ModelArts in Marketplace, and click Install.

Figure 6-1 Installation using Marketplace

🖺 Settings			
Q•	Plugins	Marketplace	Installed
> Appearance & Behavior	Q∗ ModelArts		<u>م</u> ۵
Keymap	Search Results (1)	Sort By: Relevance 🔻	
> Editor	8 8 Huawoi El ModolArte		0 0 0
Plugins 🔳	445		
> Version Control			
> Project: models			Plugin hon
> Build, Execution, Deployment			Al develop
> Languages & Frameworks			ModelArts
> Tools			With the P

NOTE

The version installed in Marketplace is the latest version.

Method 2: Install PyCharm Toolkit Using a Toolkit Package

1. Download Toolkit.

The PyCharm Toolkit package is stored in the public OBS bucket of the target site. Contact the administrator to obtain it.

PyCharm Toolkit package name: For interconnection with ModelArts Training Management of the old version, use **Pycharm-ToolKit-2.2.1.zip**. For interconnection with ModelArts Training Management of the new version, use **Pycharm-ToolKit-latest.zip**. Select a name as required.

Figure 6-2 ModelArts Training Management

Training Management					
Training Jobs ^{01d} version					
Training Jobs New					

- 2. Installing Toolkit in PyCharm
 - Install Toolkit in PyCharm by performing the following operations:
 - a. Start PyCharm on the local PC.
 - b. On the PyCharm interface, choose **File > Settings**. The **Settings** dialog box is displayed.
 - c. In the **Settings** dialog box, click **Plugins** in the left navigation pane. Click the setting icon on the right, and choose **Install Plugin from Disk**. The dialog box for selecting files is displayed.

Qr	Plugins	Marketplace	Installed	\$	
▼ Appearance & Behavior				Manage Plugin Repositories HTTP Provy Settings	
Appearance				Install Plugin from Disk	
Menus and Toolbars				Disable All Downloaded Plugins	
				Enable All Downloaded Plugins	
File Colors 🛛 🖻					
Quick Lists					
Path Variables				plugin to preview details	
Keymap					
▶ Editor					
► Version Control					
▶ Project: ModelArts-Lab-master 🛛 🐵					
Build, Execution, Deployment					
Languages & Frameworks					
► Tools					

Figure 6-3 Selecting a plug-in from the local host

- d. In the displayed dialog box, select the Toolkit package from the local directory and click **OK**.
- e. Click **Restart IDE** to restart PyCharm. In the displayed dialog box, click **Restart**.

Figure 6-4 Restarting PyCharm



f. Open a project after the restart. If the **ModelArts** tab page is displayed on the PyCharm toolbar, Toolkit has been installed.

Figure 6-5 Installation successful



6.2.2 Configuring Toolkit Using a YAML File

Configuring PyCharm Toolkit

After PyCharm is restarted, perform the following steps to configure PyCharm Toolkit:

- 1. On the PyCharm interface, choose **ModelArts > Edit Credential**. The **Edit Credential** dialog box is displayed.
- 2. Click Get more region to go to the YAML file download page.
- Go to the folder of the target region.
 Click the link on the web page to view the region information.

4. Download the YAML file to the local host.

Click the YAML file to be downloaded. The file details page is displayed. On the file details page, right-click **Raw** and choose **Save link as** from the shortcut menu to save the YAML file to the local PC.

5. In the **Edit Credential** dialog box, click **Config** to import the downloaded YAML file. After the file is imported, the message **Import successful** is displayed, indicating that the region information is configured.

Setting Domain Names and IP Addresses

Toolkit must be used by calling APIs. The API domain names of some regions are not registered. You need to configure the corresponding IP addresses and API domain names in the **hosts** file on the local PC. Generally, the **hosts** file on the local PC is stored in **C:\Windows\System32\drivers\etc**.

6.2.3 Creating Access Keys (AK and SK)

This section describes how to create access keys (AKs and SKs) on the ModelArts management console. A pair of AK and SK is used to encrypt the signature of a request, ensuring that the request is secure and integral, and that identities of the request sender and receiver are correct.

Obtaining an Access Key

- 1. On the ModelArts management console, hover the cursor over the username in the upper right corner and choose **My Credentials** from the drop-down list.
- 2. On the My Credentials page, choose Access Keys > Create Access Key.
- 3. In the **Create Access Key** dialog box that is displayed, enter the verification code received by SMS or email.
- 4. Click **OK** and save the access key file as prompted. The access key file is saved in the default download folder of the browser. Open the **credentials.csv** file to view the AK and SK.

6.2.4 Using Access Keys for Login

To connect Toolkit to ModelArts, use the access keys of the current account for login authentication.

Prerequisites

- Toolkit has been installed. If it is not installed, install it by referring to **Downloading and Installing PyCharm Toolkit**.
- The access keys of the current account have been created, and the corresponding AK and SK have been obtained. If they are not created, create them by referring to **Creating Access Keys (AK and SK)**.
- Before using Toolkit, go to the ModelArts console to configure access authorization. If the global configuration is not complete on the ModelArts management console, you cannot access and connect to ModelArts after logging in to Toolkit.

Logging In to ModelArts

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

Figure 6-6 Edit Credential

E	<u>F</u> ile	<u>E</u> dit	<u>V</u> iew	<u>N</u> avigate	<u>C</u> ode	<u>R</u> efactor	R <u>u</u> n	<u>T</u> ools	VC <u>S</u>	<u>W</u> indow		<u>H</u> elp	
	pycha	rm_pr	oject1										
ъ	Pro	niect ·	-	6	-	m –					Edit Trai	ning Job	Configuration
roje				• •							🕨 Run Trai	ning Job	
		bychar	m_pro	ject1 D:\te	st\toolk	ut\pycha							
	IIIII E	xterna	al Librai	ries									
	2 🔊	cratch	nes and	Consoles									

- 2. In the displayed dialog box, select the region where ModelArts is located, enter the AK and SK, and click **OK**.
 - **Region**: Select a region from the drop-down list.
 - Access Key ID: Enter the AK.
 - Secret Access Key: Enter the SK.
- 3. View the verification result.

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

16:01Validate Credential Success: The credential is valid.

6.3 PyCharm Toolkit (Latest Version)

6.3.1 Training a Model

6.3.1.1 Submitting a Training Job (New Version)

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

Prerequisites

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

Configuring Training Job Parameters

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

Figure 6-7 Edit training job configuration



2. In the displayed dialog box, set the training job parameters. For details about the parameters, see **Table 6-2**.

Table 6-2	Training	job	parameters
-----------	----------	-----	------------

Parameter	Description
Job Name	Name of a training job
Job Description	Brief description of a training job
Algorithm Souce	Source of the training algorithm. The options are Frequently-used and Custom .
	Frequently-used refers to the frequently-used AI engines supported by ModelArts Training Management.
	If the AI engine you use is not in the supported list, you are advised to create a training job using a custom image.
Al Engine	Select the AI engine and the version used in code. The supported AI engines are the same as the frequently- used frameworks supported by training jobs on the ModelArts management console.
Boot File Path	Training boot file. The selected boot file must be a file in the current PyCharm training project.
Code Directory	Training code directory. The system automatically sets this parameter to the directory where the training boot file is located. You can change the parameter value to a directory that is in the current project and contains the boot file.
	training code has been built in the image, this parameter can be left blank.
Image Path (optional)	URL of the SWR image

Parameter	Description
Boot Command	Command for starting a training job, for example, bash /home/work/run_train.sh python { <i>Python boot</i> <i>file and parameters</i> }. This parameter is displayed if Algorithm Source is set to Custom.
	If the command does not contain the data_url or train_url parameter, the tool automatically adds the two parameters to the end of the command when submitting the training job. The two parameters correspond to the OBS path for storing training data and the OBS path for storing training output, respectively.
Data Obs Path	OBS path for storing training data, for example, /test- modelarts2/mnist/dataset-mnist/, in which test- modelarts2 indicates a bucket name.
Training Obs Path	OBS path. A directory is automatically created in the path for storing a trained model and training logs.
Running Parameters	Running parameters. If you want to add some running parameters to your code, add them here. Separate multiple running parameters with semicolons (;), for example, key1=value1;key2=value2 . This parameter can be left blank.
Specifications	Type of resources used for training. Currently, public resource pools and dedicated resource pools are supported.
	Dedicated resource pool specifications are identified by Dedicated Resource Pool .
Compute Nodes	Number of compute nodes. If this parameter is set to 1 , the system runs in standalone mode. If this parameter is set to a value greater than 1, the distributed computing mode is used at the background.
Available/Total Nodes	When Specifications is set to a dedicated resource pool, the number of available nodes and the total number of nodes are displayed. The value of Compute Nodes cannot exceed the number of available nodes.

-			-		-	
Job Name:	MA-new-model-03-12	MA-new-model-03-12-17-12				
Job Description:						
	Frequently-used C					
	Al Engine:					-
Algorithm Source:	Boot File Path:	F:\F:\				1
	Code Directory:	F:\l _work				1
	Image Path(optional):					
Training Obs Path:						
Data Obs Path:	obs://iata/	'dataset/				
Specifications:	CPU:					
Compute Nodes:						
Running Parameters:						2 ⁷
				Analyzed Dura		
				Apply and Run	Cancel	Apply

Figure 6-8 Configuring training job parameter (public resource pool)

Figure 6-9 Configuring training job parameter (dedicated resource pool)

PC	Edit	Training	Job	Configurations	
----	------	----------	-----	----------------	--

Job Name:	MA-new-model-03-12-17	MA-new-model-03-12-17-12				
Job Description:						
	Frequently-used Cust	tom				
Algorithm Source:	Al Engine:	ensorFlow 🔻	TF-1 ,,			
	Boot File Path: F:	\D . ;ttsts.py				
	Code Directory: F:					
	Image Path(optional):					
Training Obs Path:		code/				
Data Obs Path:	obs://ra/da	staset/				
Specifications:	Dedicated Resource Poo	l: ml-cn4-regular NVIDIA-V100*4 🔻				
Compute Nodes:		Available/Total Nodes: 29 / 33 😋				
Running Parameters:						
			Apply and Run Ca	incel Apply		

×

🖺 Edit Training Job	Configurations	×
Job Name:	MA-new-model-03-12-17-15	
Job Description:		
	Frequently-used Custom	
	Image Path: swr.cn in a/tf-2.1.0:1.0.0	
Algorithm Source:	python run.py Boot Command:	
	Code Directory(optional):	
Training Obs Path:	obs://i kit/code/	
Data Obs Path:	obs:// ````````kit/data/	
Specifications:	CPU://///IGB	
Compute Nodes:		
Running Parameters:		
	Apply and Run Cancel	Apply

Figure 6-10 Configuring training job parameter (custom image)

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

NOTE

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

Figure 6-11 Training log example

	ModelArts Event Log Event Log 🌣 🗕	ModelArts Training Log job-ma-iceberg-11-22-16.0
and have	Begin to check training configuration. Begin to upload training code.	Log Name: job-ma-iceberg-11-22-16.0 💌
10	MA-iceberg-11-22-16/code/src/train_iceberg_new.py: 100% Files are uploaded successfully. Begin to get training job pre version.	INFO:Current training job status: Initializing INFO:Current training job status: Running INFO:Current training job status: Surgersful
Total Annual	Begin to create training job. Training job is created successfully.	 Restarting DNS forwarder and DHCP server dnsmasq done.
al Area	Job 1d: 411903, version 1d: 620356 Begin to get training job log.	[Modelarts Service Log]user: uid=1101(work) gid=1101(work) groups=1101(work) [Modelarts Service Log]pwd: /home/work

6.3.1.2 Stopping a Training Job

You can stop a running training job.

Stopping a Job

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

Figure 6-12 Stopping a job



6.3.1.3 Viewing Training Logs

This section describes how to view training job logs.

Viewing Training Logs in OBS

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

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

|---train-job-01 |---output |---log |---code

Viewing Training Logs in Toolkit

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

Figure 6-13 Viewing Training Logs

odelArts Training Logjob-ma-iceberg-11-22-16.0		
Log Name: job-ma-iceberg-11-22-16.0 💌		
INFO:Current training job status: Initializing INFO:Current training job status: Running INFO:Current training job status: Successful * Restarting DNS forwarder and DMCP server dnsmasg		
done. [Modelarts Service Log]user: uid=1101(work) gid=1101(work) groups=1101(v	work)	
[Modelarts Service Log]pwd: /home/work [Modelarts Service Log]app_url: 33://cnnorth1-job-test/MA-iceberg-11-22- [Modelarts Service Log]boot file: sc/train iceberg new.ov	-16/code/src/	
[Modelarts Service Log]command: src/train_iceberg_new.pydata_url=s3:/	//cnnorth1-job-test/iceberg/iceberg/	num_gpus=1train_
[Modelarts Service Log]MODELARTS_IPOIB_DEVICE: [Modelarts Service Log]dependencies_file_dir: /home/work/user-job-dir/sr [Modelarts Service log]foodelarts_create_log] modelarts_nime_found		
TNEO-root-licing Maying 11 14 0-38640055	2 Event Log	ModelArts Training Log

6.4 FAQs

6.4.1 What Should I Do If an Error Occurs During ToolKit Installation?

lssue

The following error message is displayed during ToolKit installation.

Figure 6-14 Error



Solution

This issue occurs because the plug-in version is inconsistent with the PyCharm version. You need to obtain the plug-in of the same version as the PyCharm version, that is, version 2019.2 or later.

6.4.2 An Error Occurs When You Edit a Credential in PyCharm Toolkit

Symptom

When you edit a credential in PyCharm Toolkit, the message "Validate Credential error" is displayed.



Or



Possible Causes

- Possible cause 1: Information such as the region is incorrectly configured.
- Possible cause 2: The **hosts** file is not configured or is incorrectly configured.
- Possible cause 3: The network proxy settings are incorrect.
- Possible cause 4: The AK/SK is incorrect.
- Possible cause 5: The computer time is incorrectly set.

Solution

1. Information such as the region is incorrectly configured.

Configure the correct information. For details, see **Configuring Toolkit Using a YAML File**.

For example, if the endpoint is incorrect, the authentication fails.

Incorrect example: The endpoint is preceded by https.

Figure 6-15 Configuring PyCharm Toolkit

ModelArts	
Region(s) :	cn-central-2 cn-central-2°*
	Example: region-name1 region-id1;region-name2 region-id2
Project(s):	cn-central1 cn-central-
	Example 1: keep default value. By default the project is same as region id.
	Example 2: custom projects in format region-name1[project1;region-name2[project2
ModelArts Endpoint :	https://modelartslpji.com
OBS Endpoint :	https://obs.cn-centiur درم محمد المعالية https://obs.cn-centiur
IAM Endpoint :	https://iam-pub.cn-centris.com
Console Endpoint:	default
	Endpoint Example: service-name. <region-id>.domain-name</region-id>

2. The hosts file is not configured or is incorrectly configured.

Configure the domain names and IP addresses in the **hosts** file on the local PC. For details, see **Setting Domain Names and IP Addresses**.

Network proxy settings are incorrect.

If the network requires proxy settings, check whether the proxy settings are correct. You can also use the mobile hotspot to test.

Check whether the proxy settings are correct.



Figure 6-16 PyCharm network proxy settings

4. The AK/SK is incorrect.

The obtained AK/SK is incorrect. Obtain the correct AK/SK and try again. For details, see **Creating Access Keys (AK and SK)**.

If you use a RightCloud account, contact the technical support of the region to obtain the AK/SK.

5. The computer time is incorrectly set.

Set the computer time to the correct time.

6.4.3 Why Cannot I Start Training?

If code that does not belong to the used project is selected in a boot script, training cannot be started. The following figure shows error information. You are advised to add the boot script to the project or open the project where the boot script is located, and then start the training job.

Figure 6-17 Error

Error	
x	Boot File Path must in the project directory D:\EI\ ModelArts-Lab-master\official_examples\Using_MXNet_to_Train_Caltech101\codes.
	ОК

6.4.4 What Should I Do If Error "xxx isn't existed in train_version" Occurs When a Training Job Is Submitted

Symptom

Error "xxx isn't existed in train_version" occurs when a training job is submitted. See the following figure.

Figure 6-18 Error "xxx isn't existed in train_version"



Possible Causes

The preceding error occurs because the user logs in to the ModelArts management console and deletes the training job after submitting the training job using PyCharm ToolKit.

PyCharm Toolkit records the training job IDs of ModelArts on the cloud. If you manually delete the job on the ModelArts management console, a message is displayed indicating that the job with the ID cannot be found when you submit the job locally.

Solution

If you have deleted a job on the ModelArts management console, you also need to delete the local configuration from ToolKit. To delete the local configuration, click **Edit Training Configuration**, find the job name, click the minus sign in the upper right corner, and confirm the deletion.

Figure 6-19 Deleting the local configuration

Edit Training Job Configurations		×
Job Name:	dog-cat-recognize	Delete this training job configuration
Job Description:	Frequently-used Custom	

In the displayed confirmation dialog box, confirm the information and click **Yes** to delete the configuration. After the deletion, you can create a training job configuration and submit the training job.

6.4.5 What Should I Do If an Error Occurs When I Submit a Training Job

When a training job is running, the "Invalid OBS path" error is reported.

Figure 6-20 "Invalid OBS path" error

No modified code to upload.	
Begin to get training job pre version.	
Begin to create training job.	
Http Response Code : 400	
{"is_success":false,"error_code":"ModelArts.0404","error_message":"Invalid OBS path '	//////////////////////////////////////
Create training job failed.	
ModelArts Training is Finished.	

To locate the fault, perform the following operations:

- If you are using ModelArts for the first time, log in to the ModelArts management console and complete access authorization configuration. The agency authorization mode is recommended. After the global configuration is complete, submit the job again.
- Check whether the configured Data Path in OBS exists and whether data files exist in the directory. If the directory does not exist, create a directory on OBS and upload the training data to the directory.

6.4.6 What Should I Do If an Error Occurs During Service Deployment

Before deploying a model as a service, you need to compile the configuration file and inference code based on the trained model.

If the **confi.json** configuration file or the **customize_service.py** inference code is missing in the model storage path, an error is displayed, as shown in the following figure.

Solutions:

Write the configuration file and inference code, and save them to the OBS directory where the model to be deployed resides. For details, see **Introduction to Model Package Specifications**.

Figure 6-21 Error

6.4.7 How Do I View Error Logs of PyCharm ToolKit?

The error logs of PyCharm ToolKit are recorded in the **idea.log** file of PyCharm. For example, in the Windows operating system, the path of the **idea.log** file is C:\Users\xxx\.IdeaIC2019.2\system\log\idea.log.

Search for **modelarts** in the log file to view all logs related to PyCharm ToolKit.

7 Uploading and Downloading Data in Notebook

Uploading Files to JupyterLab Downloading a File from JupyterLab to a Local Path Uploading Data from a Local IDE to a Notebook Instance Downloading Files from a Notebook Instance to a Local Directory

7.1 Uploading Files to JupyterLab

7.1.1 Scenarios

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

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

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

Optimized file uploading:

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

7.1.2 Uploading Files from a Local Path to JupyterLab

7.1.2.1 Upload Scenarios and Entries

JupyterLab provides multiple methods for uploading files.

Methods for Uploading a File

- For a file that does not exceed 100 MB, directly upload it, and details such as the file size, upload progress, and upload speed are displayed.
- For a file that exceeds 100 MB but does not exceed 5 GB, upload the file to OBS (an object bucket or a parallel file system), and then download the file from OBS to a notebook instance. After the download is complete, the file is deleted from OBS.
- For a file that exceeds 5 GB, upload it by calling ModelArts SDK or MoXing.
- For a file that shares the same name with an existing file in the current directory of a notebook instance, overwrite the existing file or cancel the upload.
- A maximum of 10 files can be uploaded at a time. The other files are in awaiting upload state. No folders can be uploaded. If a folder is required, compress it into a package, upload the package to notebook, and decompress the package in Terminal.

unzip xxx.zip # Directly decompress the package in the path where the package is stored.

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

• When multiple files are uploaded in a batch, the total number of files to be uploaded and the number of files that have been uploaded are displayed at the bottom of the JupyterLab window.

Simple 🔵 1 🖪 0 🟟 🚸 CPU: 0% | Mem: 241 / 16384 MB Uploading: 19 / 19

Prerequisites

You have used JupyterLab to open a running notebook environment.

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

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



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

Click ¹ in the navigation bar on the top of the window. In the displayed dialog box, drag or select a local file and upload it.



Figure 7-2 File upload page



7.1.2.2 Uploading a Local File Less Than 100 MB to JupyterLab

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

Figure 7-3 Uploading a file less than 100 MB

Add fi	les to Notebook			—	\times
	Uploaded files (0/1)			SELEC FILES	Ŧ
Git	Name	Status			
	🗅 typora-0-11-18-Windows.rar (67.7MB)	-	4%	7MB/s	ò
€ BS		Rows per page: 10 ▼	1-1 of 1 <	< >	>
Remote					

A message is displayed after the file is uploaded.

Figure 7-4 Uploaded					
Add fi	les to Notebook			_	×
Local	Uploaded files (1/1)		D	SELEC	
Git	Name	Status			
	urlCheck_green.zip (29.2MB)	 uploaded successfully 			
OBS		Rows per page: 10 ▼ 1-	-1 of 1 <	< >	>
Remote					

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

For a file that exceeds 100 MB but does not exceed 5 GB, upload the file to OBS (an object bucket or a parallel file system), and then download the file from OBS to the target notebook instance. After the download is complete, the file is automatically deleted from OBS.

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

Add fi	les to Notebook	- ×	
	Uploaded files (0/1)	SELECT FILES	•
Ø	Name	Status	
Git	🗅 data-set.zip (163.5MB)	1000 The file size exceeds 100MB OBS TRANSIT	
OBS PRemote		Rows per page: 10 1−1 of 1 < < > >	

Figure 7-5 Uploading a large file through OBS

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

÷.

Figure 7-6 Uploading a file through OBS

Add fi	les to Notebook	- ×
Ę	< RETURN AND CANCEL	3
Local	Set the OBS transit path	USE DEFAULT
	Enter the OBS transit path, select a path from OBS File Browser, or use the default path	CONFIRM
OBS OBS Remote	OPEN OBS FILE BROWSER 🗸 🥑	

NOTE

Set an OBS path for uploading local files to JupyterLab. After the setting, this path is used

by default in follow-up operations. To change the path, click 🥺 in the lower left corner of the file upload window.

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

Figure 7-7 Configuring an OBS path

D	RETURN AND CANCEL	í
Local	Set the OBS transit path	USE DEFAULT
Git	obs://test-	CONFIRM
	OPEN OBS FILE BROWSER 🗸	
OBS		
<i>P</i> Remote		

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

Figure 7-8 OBS File Browser

Add fi	les to Notebook				$ \times$
D	RETURN AND CAN	CEL			
Local	Set the OBS transit	path		U	ISE DEFAULT
57	obs://cn-north-7-dev			×	CONFIRM
	CLOSE OBS FILE BR	OWSER A			
(†)	obs				
-0	Back		Enter a name for query		(
Ø- Remote		Name	Last Modified	Туре	Size
	O 06024:		-	8	-
	O 06024:		-	8	-
	O automa		-	8	-
	O automa		-	8	-
	Ocn-north-7-dev		-	8	-
	O cqiaomeiyan-112		-	8	

• Method 3: Use the default path.

Figure 7-9 Using the default path to upload a file

Add files to Notebook

Local	RETURN AND CANCEL	*
	Set the OBS transit path	USE DEFAULT
57	Enter the OBS transit path, select a path from OBS File Browser, or use the default path	CONFIRM
Git	OPEN OBS FILE BROWSER 🗸	
OBS		
P Remote		

Figure 7-10 Setting an OBS path for uploading a local file

Add fi	iles to Notebook	- ×
D	RETURN AND CANCEL	
Local	Set the OBS transit path	USE DEFAULT
57	Enter the OBS transit path, select a path from OBS File Browser, or use the default path	CONFIRM
Git	OPEN OBS FILE BROWSER V	
0.00		
C Remote		
ଡ		

X

After the OBS path is set, upload a file.

Figure 7-11 Uploading a file

Add fi	les to Notebook		—	\times
Local	Uploaded files (0/1)	5	SELE	CT S
Git	Name	Status		
	D data-set.zip (163.5MB)	51	1% 50M	B/s
⊕ OBS		Rows per page: 10 ▼ 1–1 of 1		>
8 Remote				

Decompressing a package

After a large file is uploaded to Notebook JupyterLab as a compressed package, you can decompress the package in Terminal.

unzip xxx.zip # Directly decompress the package in the path where the package is stored.

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

7.1.2.4 Uploading a Local File Larger Than 5 GB to JupyterLab

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

To upload files exceeding 5 GB, upload them to OBS. Then, call the ModelArts MoXing or SDK API in the target notebook instance to read and write the files in OBS.

Figure 7-12 Uploading and downloading large files in a notebook instance



The procedure is as follows:

1. Upload the file from a local path to OBS.
- 2. Download the file from OBS to the notebook instance by calling the ModelArts SDK or MoXing API.
 - Method 1: Call the ModelArts SDK to download a file from OBS.

Example code: from modelarts.session import Session session = Session() session.obs.copy("obs://*bucket-name*/*obs_file.txt*","/home/ma-user/work/")

 Method 2: Call the ModelArts MoXing API for reading an OBS file. import moxing as mox

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

If a .zip file is downloaded, run the following command on the terminal to decompress the package:

unzip xxx.zip # Directly decompress the package in the path where the package is stored.

After the code is executed, open the terminal shown in **Figure 2** and run the **ls /home/ma-user/work** command to view the file downloaded to the notebook instance. Alternatively, view the downloaded file in the left navigation pane of Jupyter. If the file is not displayed, refresh the page.

Figure 7-13 Opening the terminal

\bowtie	File Edit View Run	Kernel Git Tabs Settings Help	
	New	Console	
_	New Launcher	Ctrl+Shift+L 🔲 Notebook	
0	Open from Path	s_ Terminal	

Figure 7-14 File downloaded to a notebook instance



Error Handling

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

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

7.1.3 Cloning an Open-Source Repository in GitHub

Files can be cloned from a GitHub open-source repository to JupyterLab.

- 1. Use JupyterLab to open a running notebook instance.
- 2. Click $\stackrel{1}{=}$ in the navigation bar on the top of the JupyterLab window. In the



displayed dialog box, click ^{Git} on the left to go to the page for cloning files from a GitHub open-source repository.

Figure 7-15 File upload icon

Add files to Notebook



Figure 7-16 Page for cloning files from a GitHub open-source repository

Auun	les to Notebook	~
L.	GitHub open-source repository clone	
Local	Enter the GitHub open-source repository URL	CLONE
Git		
∂ BS		
8 Remote		

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

GitHub open-source repository address: https://github.com/jupyterlab/ extension-examples 4.

Figure 7-17 Entering a valid address of a GitHub open-source repository

Add fi	les to Notebook	- ×
	GitHub open-source repository clone	
	https://github.com/jupyterlab/extension-examples Branch: master -	
Oit	Files preview	
() OBS	 .github command-palette 	
8 Remote	commands	
View t	he clone process.	

Figure 7-18 Process of cloning a repository

Add files to Notebook	—	\times

	extension-examples is being cloned
	C
	Begin clone
lono	

5. Complete the clone.

Error Handling

- Failing to clone the repository may be caused by network issues. In this case, run the **git clone https://github.com/jupyterlab/extension-examples.git** command on the **terminal** page to test the network connectivity.
- If the repository already exists in the current directory of the notebook instance, the system displays a message indicating that the repository name already exists. In this case, you can overwrite the existing repository or click
 - to cancel the cloning.

7.1.4 Uploading OBS Files to JupyterLab

In JupyterLab, you can download files from OBS to a notebook instance.

- 1. Use JupyterLab to open a running notebook instance.
- 2. Click ¹ in the navigation bar on the top of the JupyterLab window. In the

displayed window, click on the left to go to the OBS file upload page.

Figure 7-19 File upload icon

+	Ð	±	C	\mathfrak{H}^{+}
Name			•	Last Modified
• 🖪 Untitled.	ipynb			an hour ago

Figure 7-20 OBS file upload

Add fi	les to Notebook	—	\times
L.	OBS file upload		
Local	Enter the OBS file path, or select a path from OBS File Browser	U	PLOAD
Git	OPEN OBS FILE BROWSER V		
OBS			
& Remote			

- 3. Set an OBS file path in either of the following ways:
 - Method 1: Enter a valid OBS file path in the text box and click **Upload**.

Figure 7-21 Entering a valid OBS file path Add files to Notebook

D,	OBS file upload		
Local	obs://modelart	\times	UPLOAD
Git	OPEN OBS FILE BROWSER 🗸		
∂ OBS			
& Remote			

NOTE

Enter an OBS file path instead of a folder path. Otherwise, the upload fails.

• Method 2: Open OBS File Browser, select an OBS file path, and click Upload.

 \times

Figure 7-22 Uploading an OBS File

OBS fi	le upload		
obs://(be00d5092fbdc0013d201342/f9937af	ia-26cb-4a1e-a002-a376897dbbbc-2022-07-28-15-43-4 🗙	UPLOA
CLOSE			
obs /	304be00d5 / f9937afa-26	6cb-4a	
Ва	ck	Enter a name for query	
			0:
	Name	Last Modified Type	Size
 ideal 	Name	Last Modified Type Thu, 28 Jul 2022 07:43:45 G	686MB

Error Handling

Files may not be uploaded successfully.

Possible causes:

- The OBS path is set to a folder instead of a file path.
- The file in OBS is encrypted. In this case, go to the OBS console and check whether the file is encrypted.
- The OBS bucket and notebook instance are not in the same region. In this case, ensure that the target OBS bucket and notebook instance are in the same region. If the OBS bucket and notebook instance are in different regions, the access to OBS is denied.
- The account does not have the permission to access the OBS bucket. In this case, ensure that the notebook account has the permission to read data in the OBS bucket.

7.1.5 Uploading Remote Files to JupyterLab

Files can be downloaded through remote file addresses to JupyterLab.

Method: Enter the URL of a remote file in the text box of a browser, and the file is directly downloaded.

- 1. Use JupyterLab to open a running notebook instance.
- 2. Click ¹ in the navigation bar on the top of the JupyterLab window. In the

displayed window, click \mathscr{P} on the left to go to the remote file upload page.

Figure 7-23 File upload icon

+		1	G	\$ ⁺
Name		*		Last Modified
• 🖪 Untitled.	ipynb			an hour ago

Figure 7-24 Remote file upload page

Add f	iles to Notebook	$ \times$
Ę	Remote file upload	
Local	Enter the remote file URL	UPLOAD
Git	() The URL can be used to download files directly from the browser. For example: http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz	
⊕ OBS		
8 Remote		

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

Figure 7-25 Entering a valid remote file URL

Add files to Notebook

http://	nages-idx3-ubyte.gz	×
The file you want to upl	oad is train-images-idx3-ubyte.gz	

Error Handling

Failing to upload the remote file may be caused by network issues. In this case, enter the URL of the remote file in the text box of a browser to check whether the file can be downloaded.

7.2 Downloading a File from JupyterLab to a Local Path

Files created in JupyterLab can be downloaded to a local path. The operations for downloading a file are the same, regardless of whether the created notebook instance uses the default or EVS storage.

- If a file is less than or equal to 100 MB, directly download it from JupyterLab. For details, see **Downloading a File Less Than or Equal to 100 MB**.
- If a file is larger than 100 MB, use OBS to transfer it to your local path. For details, see **Downloading a File Larger Than 100 MB**.

 \times

Downloading a File Less Than or Equal to 100 MB

In the JupyterLab file list, right-click the file to be downloaded and choose **Download** from the shortcut menu. The file is downloaded to your browser's downloads folder.

Name		Last Modi	ified
Pytorch_1	Open Open With Open With Open in New Browser Tab Rename Delete Cut Cut Copy Duplicate Download Shut Down Kernel Copy Path New Folder New File New Markdown File Paste		ago

Figure 7-26 Downloading a file

Downloading a File Larger Than 100 MB

Use OBS to transfer the file from the target notebook instance to the local path. To do so, perform the following operations:

 In the notebook instance, create an IPYNB file larger than 100 MB and use MoXing to upload it to OBS. Example code is as follows: import moxing as mox mox.file.copy('/home/ma-user/work/obs file.txt', 'obs://bucket_name/obs_file.txt')

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

- 2. Use OBS or ModelArts SDK to download the file from OBS to the local path.
 - Method 1: Use OBS to download the file.
 - Download **obs_file.txt** from OBS to the local path. If a large amount of data is to be downloaded, use OBS Browser+ to download.

- Method 2: Use ModelArts SDK to download the file.
 - i. Download and install the SDK locally.
 - ii. Authenticate sessions.
 - iii. Download the file from OBS to the local path. Example code is as follows:

```
from modelarts.session import Session
session=Session(access_key='***',secret_key='***',project_id='***',region_name='***')
session.download_data(bucket_path="/bucket_name/obs_file.txt",path="/home/user/
obs_file.txt")
```

7.3 Uploading Data from a Local IDE to a Notebook Instance

If the data is less than or equal to 500 MB, directly copy the data to the local IDE.

If the data is larger than 500 MB, upload the code to OBS and then to the EVS disk associated with the target notebook instance.

- 1. Upload data to OBS.
- 2. Call the **mox.file.copy_parallel** MoXing API provided by ModelArts in the terminal of the local IDE to transfer data from OBS to EVS of the notebook instance.

Figure 7-27 Uploading data to a notebook Instance through OBS



The following shows how to enable terminal in PyCharm.



Figure 7-28 Enabling the terminal in PyCharm

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

Manually access the development environment. cat /home/ma-user/README # Select the source environment. source /home/ma-user/miniconda3/bin/activate MindSpore-python3.7-aarch64 # Use MoXing for access. import moxing as mox # Download a folder from OBS to EVS. mox.file.copy_parallel('obs://bucket_name/sub_dir_0', '/tmp/sub_dir_0')

7.4 Downloading Files from a Notebook Instance to a Local Directory

Files created in Notebook can be downloaded to a local path. The operations for downloading a file are the same, regardless of whether the created notebook instance uses the default or EVS storage.

Downloading Files from a Notebook Instance to a Local Directory in PyCharm

In the **Project** directory of the local IDE, right-click and choose **Deployment** from the shortcut menu. Click **Download from xxx** (notebook instance name) to download the Notebook2.0 project file to the local PC.

Figure 7-29 Downloading files from a notebook instance to a local directory in PyCharm

