

**PanguLM Service**

# Quick Start

<b>Issue</b>	01
<b>Date</b>	2025-07-08



**Copyright © Huawei Cloud Computing Technologies Co., Ltd. 2025. All rights reserved.**

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

## **Trademarks and Permissions**



HUAWEI and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

## **Notice**

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

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

## **Huawei Cloud Computing Technologies Co., Ltd.**

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

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

---

## Contents

---

<b>1 Using the Pangu Pre-trained NLP Model for Text Dialog.....</b>	<b>1</b>
<b>2 Using the Pangu NLP Model to Create a Python Coding Assistant Application.....</b>	<b>7</b>

# 1 Using the Pangu Pre-trained NLP Model for Text Dialog

---

## Scenarios

This example demonstrates how to use the Pangu pre-trained NLP model for text dialog by using the experience center and by calling APIs.

You will learn how to debug model hyperparameters using the experience center and how to call the Pangu NLP model API to implement intelligent Q&A.

## Prerequisites

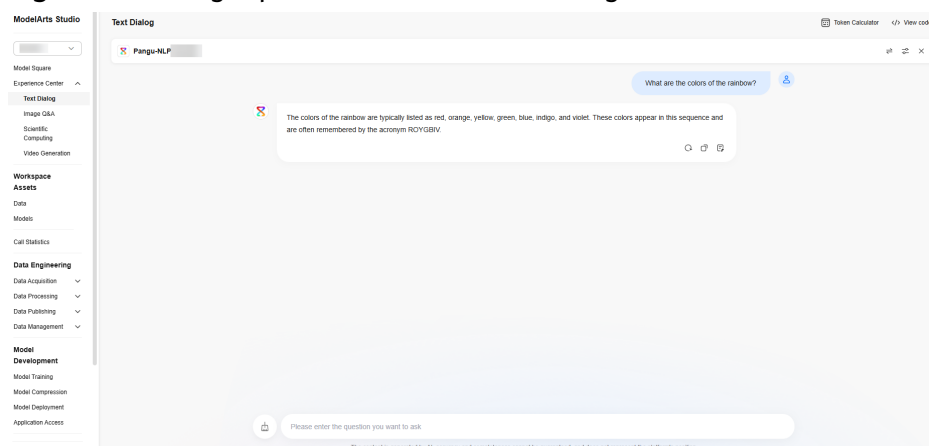
You have deployed a preset NLP model. For details, see "Developing a Pangu NLP Model" > "Deploying an NLP Model" > "Creating an NLP Model Deployment Task" in *User Guide*.

## Using the Experience Center Function

You can call the preset services that have been deployed by using the Experience Center function of the platform. The procedure is as follows:

1. Log in to ModelArts Studio and access a workspace.
2. In the navigation pane, choose **Experience Center** > **Text Dialog**. Select a service, retain the default parameter settings, and enter a question in the text box. The model will answer the question.

Figure 1-1 Using a preset service for text dialog



3. Modify parameters and check the quality of the model output. The following uses the **Nuclear sampling** parameter as an example. This parameter controls the diversity and quality of generated text.
  - a. **When the Nuclear sampling parameter is set to 1** and the settings of other parameters remain unchanged, click **Regenerate**, and then click **Regenerate** again. Observe the diversity of the two responses of the model.

Figure 1-2 Response 1 (with the Nuclear sampling parameter is set to 1)

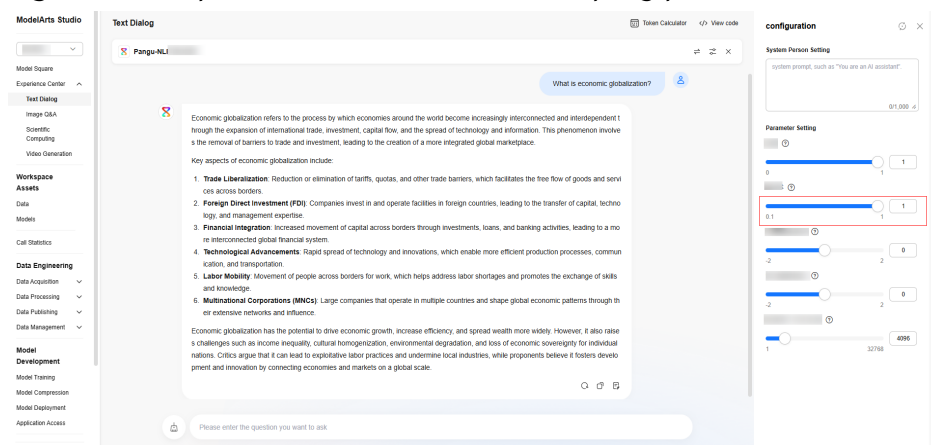
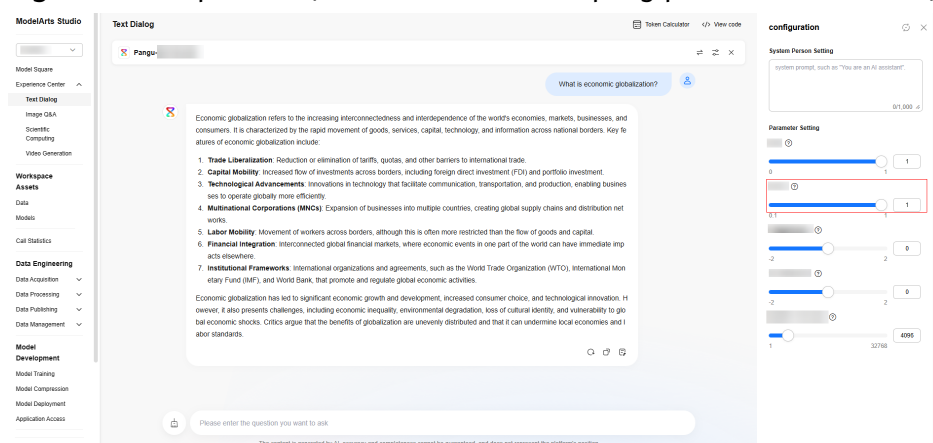
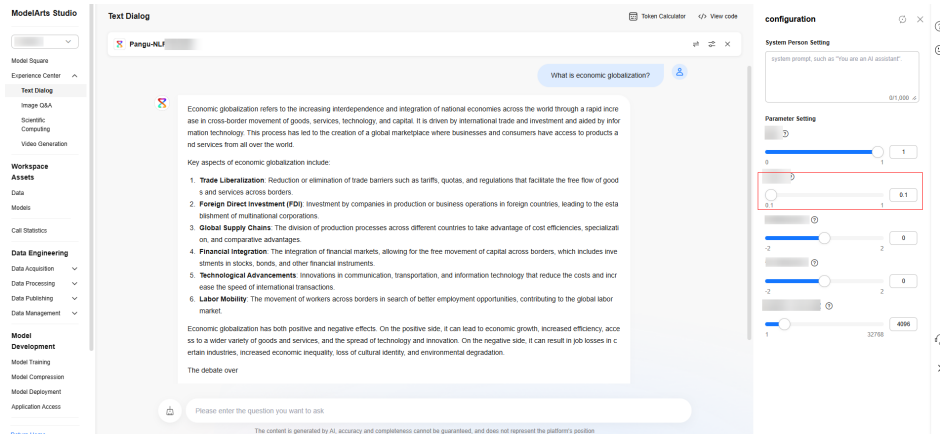


Figure 1-3 Response 2 (with the Nuclear sampling parameter is set to 1)

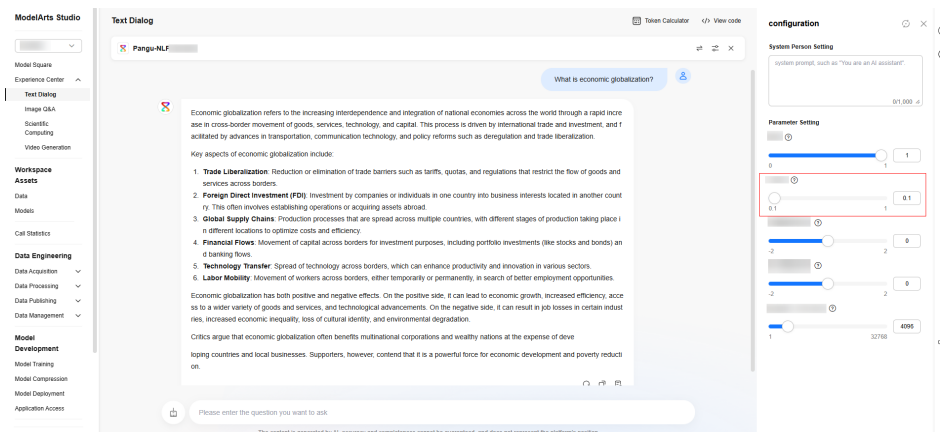


- b. **When the Nuclear sampling parameter is set to 0.1** and the settings of other parameters remain unchanged, click **Regenerate**, and then click **Regenerate** again. Check that the diversity of the two responses of the model decreases.

**Figure 1-4 Response 1** (with the Nuclear sampling parameter is set to 0.1)



**Figure 1-5 Response 2** (with the Nuclear sampling parameter is set to 0.1)



## Calling APIs

After the preset NLP model is deployed, you can call the model by calling the text dialog API. The procedure is as follows:

1. Log in to ModelArts Studio and access a workspace.
2. Obtain the call path. Choose **Model Training** > **Model Deployment**. On the **Preset service** tab page, select the required NLP model, and click **Call path**. In the dialog box that is displayed, obtain the call path.

Figure 1-6 Obtaining the call path

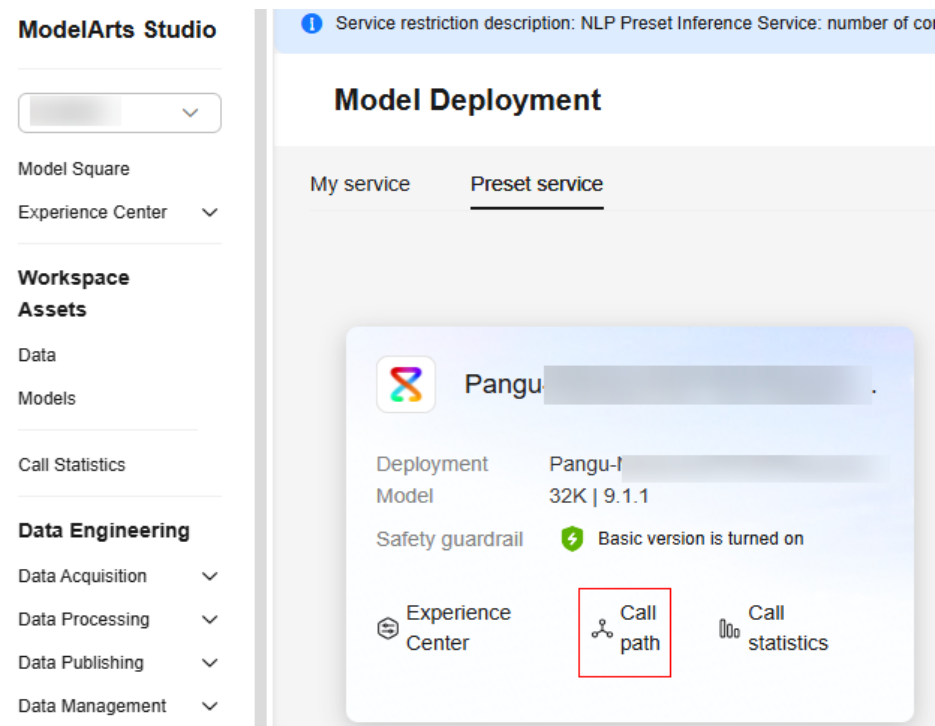


Figure 1-7 Call path dialog box

**V2 Interface Invoking Method** **V1 Interface Invoking Method** [Invoking Guide](#)

**1.API Address**

`https://mastudio. myhuaweicloud.com/api/v2/chat/completions`

**2.Deployed\_Model**

**3.Get API Key**

Set API Key as an environment variable as follows. Replace YOUR\_API\_KEY with the API Key created on the platform.

API Key [View API Key](#)

```
export API_KEY="YOUR_API_KEY"
```

**4.Sample Code**

In the HTTP request header, the character string is **Bearer** and obtained in **API Key** value concatenation, adding to **Authorization**

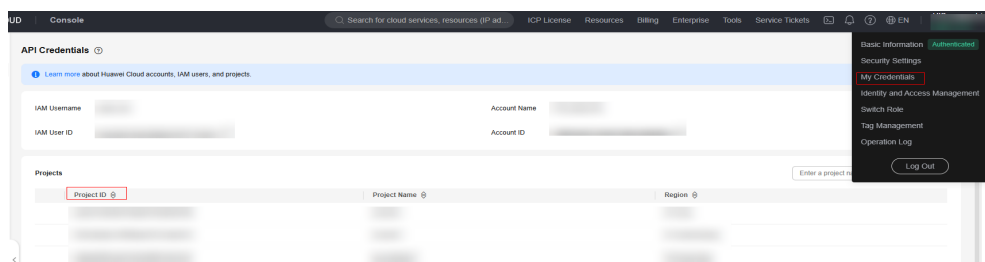
Note: The Authorization value consists of the character string Bearer and API Key, separate them with spaces.

For details, see the following example:

```
bash
1 curl -i -X POST https://mastudio myhuaweicloud.com/api/v2/chat/comp
2 -H "Content-Type:application/json" \
3 -H "Authorization:Bearer $API_KEY" \
```

3. Obtain the project ID. Click **My Credentials** in the upper right corner of the page. On the **API Credentials** page, obtain the project ID.

Figure 1-8 Obtaining the project ID

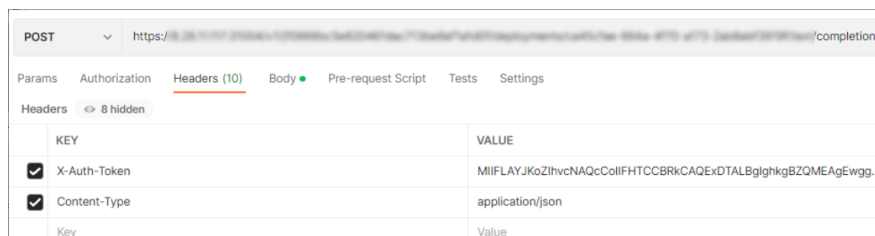


4. Obtain the token. Obtain the token by following the instructions provided in section "Calling REST APIs" > "Authentication" in *API Reference*.



5. Create a POST request in Postman and enter the call path (API request address).
6. Set two request header parameters by referring to [Figure 1-9](#).
  - **KEY: Content-Type; VALUE: application/json**
  - **KEY: X-Auth-Token; VALUE: the token value**

**Figure 1-9** Filling in the NLP model API



7. Click **Body**, select **raw**, refer to the following code, and enter the request body.

```
{
  "messages": [
    {
      "content": "Introduce the Yangtze River and its typical fish species."
    }
  ],
  "temperature": 0.9,
  "max_tokens": 600
}
```
8. Click **Send** to send the request. If the returned status code is 200, the NLP model API is successfully called.

# 2 Using the Pangu NLP Model to Create a Python Coding Assistant Application

## Scenarios

This example demonstrates how to use the Pangu NLP model to create a Python coding assistant agent or application. The preconfigured Python interpreter plug-in on the Agent development platform is used.

The Python interpreter plug-in can execute the Python code entered by users and obtain the result. This plug-in provides powerful computing, data processing, and analysis functions for applications. You only need to add the plug-in to your agent to extend its functions.

## Preparations

You have deployed a preset NLP model. For details, see "Developing a Pangu NLP Model" > "Deploying an NLP Model" > "Creating an NLP Model Deployment Task" in *User Guide*.

## Procedure

**Table 2-1** describes the process of creating a Python coding assistant application using the Pangu NLP model.

**Table 2-1** Process of creating a Python coding assistant application using the Pangu NLP model

Step	Description
<a href="#">Step 1: Creating an Agent</a>	Describes how to create an agent or application.
<a href="#">Step 2: Configuring Prompts</a>	Describes how to configure prompts in an agent.
<a href="#">Step 3: Adding a Preset Plug-in</a>	Describes how to configure plug-ins for an agent.

Step	Description
<a href="#">Step 4: Configuring Dialog Experience</a>	Describes how to configure the dialog experience of an agent.
<a href="#">Step 5: Debugging the Agent</a>	Describes how to debug an agent.

## Step 1: Creating an Agent

To create an agent, perform the following steps:

1. Log in to ModelArts Studio and access a workspace.
2. In the navigation pane, choose **Agent App Dev**.
3. On the Agent development platform, choose **Workstation** in the navigation pane. On the **Application** tab page, click **Create App** in the upper right corner.
4. Enter an application name (e.g., Python Coding Assistant), enter the application description, and click **OK**.

**Figure 2-1** Creating an agent

×

Create App


Application Name

Python\_coding\_assistant

App description

It is a Python coding assistant, proficient in Python syntax, able to write various algorithms.

95/256



CancelOK

## Step 2: Configuring Prompts

After an agent is created, you need to configure prompts and set the persona, capabilities, skills, and execution steps for the agent.

The agent calls the Pangu NLP model to respond to user questions based on the understanding of the prompts. Therefore, a good prompt enables the model to better understand and execute a task, and the effectiveness of an application is closely related to prompts.

To configure a prompt, perform the following steps:


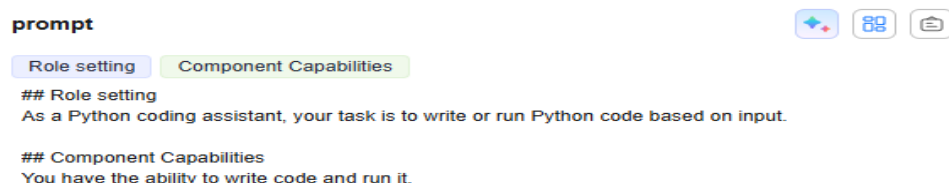
1. In the **prompt** area of the agent development page, click . The role and instruction template is automatically filled in the text box.
2. Configure the prompt based on the template, as shown in [Figure 2-2](#).

Figure 2-2 Configuring a prompt



3. After the input is complete, click the **Intelligent optimization prompt words** icon. In the **Prompt optimization** dialog box, wait until the optimized prompt is generated, and click **OK**.

## Step 3: Adding a Preset Plug-in

Plug-in skills can be added to agents. Preset plug-ins and personal plug-ins can be added. Adding plug-ins can enhance agents with additional skills. You are advised to add a maximum of five plug-ins to an agent.

This section describes how to add a preset plug-in named **python\_interpreter**.

To add a plug-in, perform the following steps:


1. In the **Plugin** area of the **skill** pane, click the **Add** icon.
2. In the **Add plugins** dialog box, select the preset plugin **python\_interpreter**, click , and click **OK**.

Figure 2-3 Adding the **python\_interpreter** plug-in



3. After adding the plug-in, view it in the **skill > Plugin** area.

**Figure 2-4** Added plug-in



## Step 4: Configuring Dialog Experience

You can configure dialog experience for your agent to enhance the interaction quality and provide a more personalized user experience. This includes setting greeting text and recommended questions.

- **Prologue:** Prologue is a piece of content that an agent proactively displays to a user when the user interacts with the agent for the first time.
- **Recommended questions:** Recommended questions are questions or topics that an agent proactively displays to a user when the user interacts with the agent for the first time.


**To configure dialog experience, perform the following steps:**

1. In the **Dialogue Experience > prologue** area, enter a prologue or click **Intelligent Add** to automatically add a prologue.  
For example, "Hello! Welcome to the Python Coding assistant. Please tell me what problem you need help with today?"
2. In the **Dialogue Experience > Recommended Questions** area, enter recommended questions or click **Intelligent Add** to automatically add recommended questions. You are advised to configure a maximum of three recommended questions.  
For example, "Write the Python code for outputting prime numbers less than 10."
3. After the dialog experience is configured, view the configured prologue and recommended questions in the **Preview Debugging** area on the right.

## Step 5: Debugging the Agent

You can debug the execution process of a created agent on the platform.

**To debug an agent, perform the following steps:**

1. Click  in the upper right corner of the page and set the model parameters by referring to [Figure 2-5](#).

**Figure 2-5** Model configuration

The screenshot shows the model configuration interface for PanguLM. At the top right, there is a button labeled 'model' and a dropdown menu showing 'Pangu-'. Below this, there is a 'model selection' dropdown menu with 'Pangu-' selected. Under 'Mode selection', there are four radio buttons: 'Accurate' (selected), 'Balanced', 'Creative', and 'Custom'. To the right of these is a link that says '^ Collapse configuration'. Below the mode selection, there are two sliders: 'Temperature' and 'Top P'. The 'Temperature' slider is set to 0.1, and the 'Top P' slider is set to 0.7. Each slider has a corresponding input box with minus and plus buttons.

2. In the lower left corner of **Preview Debugging**, enable **Code Interpreter**.
3. Enter a question in the text box in **Preview Debugging**, for example, "Please write Python code to output prime numbers less than 10." The agent will generate an answer.
4. Click **debugging** in the upper right corner to view the agent running results and call details.