ModelArts

Contenedores de Docker con ModelArts

Edición 01

Fecha 2023-03-02

HUAWEI TECHNOLOGIES CO., LTD.

Marcas y permisos


y otras marcas registradas de Huawei pertenecen a Huawei Technologies Co., Ltd.
Todas las demás marcas registradas y los otros nombres comerciales mencionados en este documento son propiedad de sus respectivos titulares.

Aviso

Las funciones, los productos y los servicios adquiridos están estipulados en el contrato celebrado entre Huawei y el cliente. Es posible que la totalidad o parte de los productos, las funciones y los servicios descritos en el presente documento no se encuentren dentro del alcance de compra o de uso. A menos que el contrato especifique lo contrario, ninguna de las afirmaciones, informaciones ni recomendaciones contenidas en este documento constituye garantía alguna, ni expresa ni implícita.

La información contenida en este documento se encuentra sujeta a cambios sin previo aviso. En la preparación de este documento se realizaron todos los esfuerzos para garantizar la precisión de sus contenidos. Sin embargo, ninguna declaración, información ni recomendación contenida en el presente constituye garantía alguna, ni expresa ni implícita.

Huawei Technologies Co., Ltd.

Dirección: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Sitio web: https://www.huawei.com

Email: support@huawei.com

Índice

1 Introducción a las imágenes personalizadas

Las imágenes de uso frecuente están preestablecidas en ModelArts. Sin embargo, si tiene algunos requisitos especiales para motores de aprendizaje profundo o bibliotecas de desarrollo, las imágenes preestablecidas no pueden cumplir con sus requisitos. Para resolver este problema, ModelArts permite la personalización de imágenes para que pueda personalizar los motores de tiempo de ejecución.

ModelArts se ejecuta en contenedores. Las imágenes personalizadas son las imágenes de contenedores personalizadas que se ejecutan en ModelArts. Las imágenes personalizadas admiten los parámetros de CLI y variables de entorno en formato de texto libre, con una gran flexibilidad para una amplia gama de motores de cómputo.

Servicios vinculados

El uso de una imagen personalizada puede implicar los siguientes servicios:

- SWR

  Software Repository for Container (SWR) proporciona una gestión fácil, segura y confiable de las imágenes de contenedores a lo largo de su ciclo de vida, facilitando el despliegue de las aplicaciones en contenedores. Puede cargar, descargar y gestionar las imágenes de contenedores a través de la consola de SWR, las API de SWR o la CLI de la comunidad.

  Las imágenes personalizadas utilizadas por ModelArts para entrenar o crear aplicaciones de IA se obtienen de la lista de gestión de servicios de SWR. Sus imágenes personalizadas deben subirse a SWR.

  Figura 1-1 Obtención de imágenes

- OBS

  Object Storage Service (OBS) es un servicio de almacenamiento en la nube optimizado para almacenar las cantidades masivas de datos. Proporciona capacidades de almacenamiento ilimitadas, seguras y altamente confiables con un costo relativamente bajo.

ModelArts intercambia datos con OBS. Puede almacenar datos en OBS.

- ECS

  Un Elastic Cloud Server (ECS) es una unidad informática básica que consta de vCPUs, memoria, sistema operativo (SO) y discos de Elastic Volume Service (EVS). Después de crear un ECS, puede usarlo de manera similar a cómo usaría su PC local o servidor físico.

  Puede crear una imagen personalizada en las instalaciones o en un ECS.

  NOTA

  Cuando utiliza una imagen personalizada, es posible que ModelArts deba acceder a los servicios dependientes, como SWR y OBS. La imagen personalizada solo se puede usar después de que se autorice el acceso a estos servicios dependientes. Es una buena práctica utilizar una delegación para la autorización. Una vez configurada la delegación, los permisos para acceder a los servicios dependientes se delegan en ModelArts para que ModelArts pueda utilizar los servicios dependientes y realizar operaciones en los recursos en su nombre. Para obtener más información, consulte la sección **Configuración de la autorización de acceso**.

Escenarios de aplicación de imágenes personalizadas de ModelArts

- **Usar imágenes personalizadas en las instancias de notebook**

  Si las imágenes preestablecidas de las instancias de notebook no pueden cumplir los requisitos, puede crear una imagen personalizada instalando y configurando el software y otros datos requeridos por el entorno en una imagen preestablecida. A continuación, utilice la imagen personalizada para crear nuevas instancias de notebook.

- **Usar una imagen personalizada para crear trabajos de entrenamiento**

  Si ha desarrollado un modelo o script de entrenamiento localmente pero ModelArts no admite el motor de IA, cree una imagen personalizada y súbala a SWR. A continuación, utilice esta imagen para crear un trabajo de entrenamiento sobre ModelArts y utilice los recursos proporcionados por ModelArts para entrenar modelos.

- **Usar una imagen personalizada para crear aplicaciones de IA**

  Si ha desarrollado un modelo utilizando un motor de IA que no es compatible con ModelArts, para utilizar este modelo para crear aplicaciones de IA, haga lo siguiente: Cree una imagen personalizada, importe la imagen a ModelArts y utilícela para crear aplicaciones de IA. Las aplicaciones de IA creadas de esta manera pueden gestionarse de forma centralizada y desplegarse como servicios.

2 Uso de imágenes personalizadas en instancias de notebook

2.1 Descripción general

Durante el desarrollo y el tiempo de ejecución de los servicios de IA, es necesario depurar las complejas dependencias de entorno para solidificar las configuraciones. En las mejores prácticas de desarrollo de IA de ModelArts se utilizan imágenes de contenedores para solidificar el entorno de tiempo de ejecución. De esta manera, las dependencias se pueden gestionar y el entorno de tiempo de ejecución se puede cambiar fácilmente. Los recursos de contenedores proporcionados por ModelArts permiten el desarrollo rápido y eficiente de IA y la iteración de experimentos de modelos.

ModelArts proporciona un grupo de imágenes preestablecidas de forma predeterminada. Estas imágenes ofrecen las siguientes características:

1. Listo para usar sin ninguna configuración: En ciertos escenarios, los entornos dependientes para el desarrollo de IA se solidifican para proporcionar las configuraciones óptimas de software, de SO y de red. Se han probado completamente en el hardware de destino para garantizar la compatibilidad y el rendimiento óptimos.
2. Configuración personalizable: Las imágenes preestablecidas se almacenan en el repositorio de SWR. Personalice las configuraciones de las imágenes preestablecidas y regístrelas como las imágenes personalizadas.
3. Seguros y confiables: las políticas de acceso, la asignación de permisos de usuario, el análisis de vulnerabilidades para el software de desarrollo y el SO están reforzadas con la seguridad basada en las mejores prácticas para garantizar la seguridad de las imágenes.

Si las imágenes preestablecidas no pueden cumplir sus requisitos de servicio, cree una imagen personalizada.

2.2 Guardar una instancia de Notebook como una imagen personalizada

2.2.1 Creación de una instancia de notebook mediante una imagen predefinida

Escenarios de aplicación

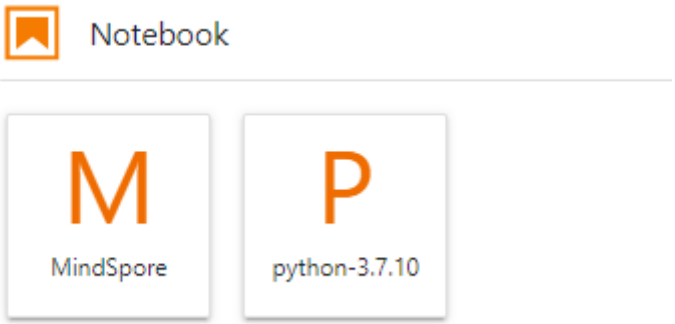
ModelArts DevEnviron proporciona un grupo de imágenes preestablecidas. Puede utilizar una imagen preestablecida para crear una instancia de notebook. Después de personalizar las configuraciones de la instancia, cree una imagen personalizada. A continuación, puede utilizar directamente la imagen de ModelArts para los trabajos de entrenamiento sin ninguna adaptación.

Las versiones de las imágenes preestablecidas de ModelArts se proporcionan en función de los comentarios del usuario y la estabilidad de la versión. Si su desarrollo se puede llevar a cabo utilizando versiones predefinidas de ModelArts, por ejemplo, MindSpore 1.5, utilice las imágenes predefinidas. Estas imágenes han sido completamente verificadas y tienen muchos paquetes de instalación de uso común incorporados. Están listos para usar, lo que le libera de configurar el entorno.

Características

Las imágenes preestablecidas en el DevEnviron de ModelArts incluyen:

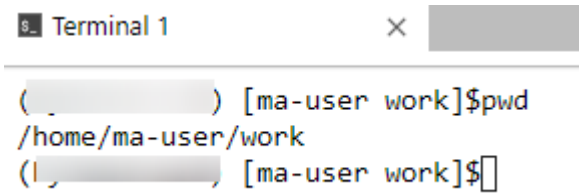
- Paquetes preestablecidos comunes: Los motores comunes de IA basados en el estándar Conda, los paquetes comunes de software de análisis de datos como Pandas y Numpy, y el software de herramientas comunes como CUDA y CUDNN se incluyen para cumplir con los requisitos comunes de desarrollo de IA.
- Entornos Conda preestablecidos: Se crean un entorno Conda y Conda Python básico (excluyendo cualquier motor de IA) para cada imagen preestablecida. La siguiente figura muestra el entorno de Conda para una imagen MindSpore preestablecida.



  Seleccione un entorno de Conda basado en si se utiliza MindSpore para la depuración.
- Notebook: una aplicación web que permite codificar en la GUI y combinar el código, las ecuaciones matemáticas y el contenido visualizado en un documento.
- Complementos de JupyterLab: permiten cambiar la variante y la detención de la instancia para mejorar la experiencia del usuario. Después de detener una instancia de notebook, sus CPU y memoria ya no se facturan.
- SSH remoto: le permite iniciar y depurar remotamente una instancia de notebook desde un PC local.
- Después de que las imágenes preestablecidas en ModelArts DevEnviron admitan la personalización, las imágenes personalizadas se pueden utilizar directamente en ModelArts para trabajos de entrenamiento.

Una imagen preestablecida ModelArts se inicia como el usuario **ma-user**. **/home/ma-user/work** es el directorio predeterminado de trabajo de una instancia de notebook a la que se accede.

Después de crear una instancia, su almacenamiento persistente se monta en **/home/ma-user/work**. Los datos almacenados en este directorio se conservan incluso después de que la instancia se detenga o reinicie. Sin embargo, los datos almacenados en otros directorios no se conservarán. Cuando utilice un entorno de desarrollo, almacene datos persistentes en **/home/ma-user/work**.



Uso de una imagen preestablecida

Cuando cree una instancia de notebook, seleccione una imagen preestablecida. Después de crear la instancia del notebook, acceda a ella y utilícela.

1. Inicie sesión en la consola de gestión ModelArts. En el panel de navegación de la izquierda, elija **DevEnviron > Notebook** para cambiar a la página de **Notebook** de nueva versión.
2. Haga clic en **Create**. En la página **Create Notebook**, seleccione una imagen pública, configure otros parámetros y envíela para su creación. Para obtener más información sobre los parámetros, consulte .
3. Cuando se esté ejecutando la instancia del notebook, acceda a la instancia del notebook para utilizar la imagen seleccionada.

2.2.2 Instalación de bibliotecas externas en una instancia de notebook

Instale paquetes de dependencias de desarrollo en una instancia de notebook para facilitar el uso de pip y Conda. La fuente pip se ha configurado y se puede utilizar directamente para la instalación. La fuente Conda necesita más configuraciones.

Esta sección describe cómo configurar el origen de Conda en una instancia de notebook.

Configuración del origen de Conda

El software Conda ha sido preestablecido en imágenes.

Comandos comunes de Conda

Para más detalles sobre todos los comandos de Conda, consulte los **documentos oficiales de Conda**. En la siguiente tabla solo se enumeran los comandos comunes.

Tabla 2-1 Comandos comunes de Conda

Descripción	Comando
Obtener ayuda en línea.	`conda --help` `conda update --help # Obtain help for a command, for example, update.`
Ver la versión de Conda.	`conda -V`
Actualizar Conda.	`conda update conda  # Update Conda.` `conda update anaconda # Update Anaconda.`

Gestionar los entornos.	`conda env list  # Show all virtual environments.`
	`conda info -e # Show all virtual environments.`
	`conda create -n myenv python=3.7 # Create an environment named myenv with Python version 3.7.`
	`conda activate myenv  # Activate the myenv environment.`
	`conda deactivate  # Disable the current environment.`
	`conda remove -n myenv --all # Delete the myenv environment.`
	`conda create -n newname --clone oldname # Clone the old environment to the new environment.`

Gestionar los paquetes.	`conda list  # Check the packages that have been installed in the current environment.`
	`conda list  -n myenv  # Specify the packages installed in the myenv environment.`
	`conda search numpy # Obtain all information of the numpy package.`
	`conda search numpy=1.12.0 --info  # View the information of NumPy 1.12.0.`
	`conda install numpy pandas  # Concurrently install the NumPy and Pandas packages.`
	`conda install numpy=1.12.0  # Install NumPy of a specified version.`
	`# The install, update, and remove commands use -n to specify an environment, and the install and update commands use -c to specify a source address.`
	`conda install -n myenv numpy  # Install the numpy package in the myenv environment.`
	`conda install -c https://conda.anaconda.org/anaconda numpy  # Install NumPy using https://conda.anaconda.org/anaconda.`
	`conda update numpy pandas   # Concurrently update the NumPy and Pandas packages.`
	`conda remove numpy pandas   # Concurrently uninstall the NumPy and Pandas packages.`
	`conda update --all # Update all packages in the current environment.`

Eliminar Conda.	`conda clean -p      # Delete useless packages.`
	`conda clean -t      # Delete compressed packages.`
	`conda clean -y --all # Delete all installation packages and clear caches.`

Guardar como una imagen

Después de instalar las librerías externas, guarde el entorno utilizando la función de guardado de imágenes proporcionada por notebook de ModelArts de la nueva versión. Puede guardar una instancia de notebook en ejecución como una imagen personalizada con un clic singular para su uso en el futuro. Después de instalar los paquetes de dependencias en una instancia de notebook, es una buena práctica guardar la instancia como una imagen para evitar que se pierdan los paquetes de dependencias. Para obtener más información, véase **Guardar una imagen de entorno de Notebook**.

2.2.3 Guardar una imagen de entorno de Notebook

Para guardar una imagen de entorno de notebook, haga los pasos siguientes: Cree una instancia de notebook con una imagen preestablecida, instale el software personalizado y las dependencias en la instancia y guarde la instancia en ejecución como una imagen contenedora.

En la imagen guardada, se conserva el paquete de dependencias instalado, pero los datos almacenados en el **home/ma-user/work** para el almacenamiento persistente no se almacenarán. En el desarrollo remoto a través de VS Code, los complementos instalados en el servidor se conservan en la imagen guardada.

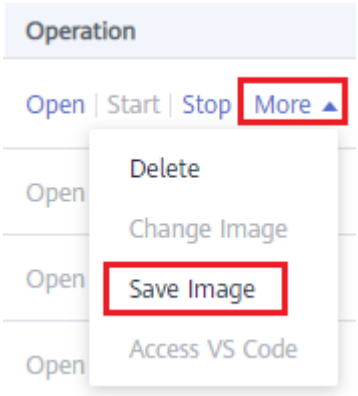
Prerrequisitos

La instancia del notebook de destino está ejecutando.

Guardar una imagen

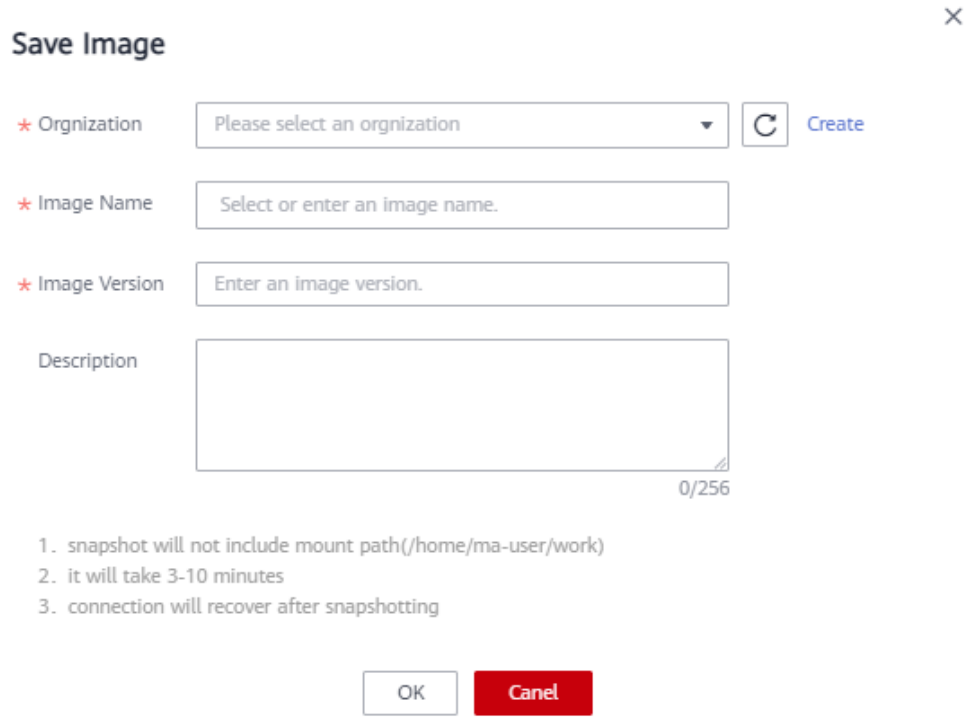
1. Inicie sesión en la consola de gestión de ModelArts y elija **DevEnviron** > **Notebook** en el panel de navegación de la izquierda para cambiar a notebook de la nueva versión.
2. En la lista de instancias de notebook, seleccione la instancia de notebook de destino y elija **Save Image** en la lista desplegable **More** de la columna **Operation**. Aparece el cuadro de diálogo **Save Image**.

   Figura 2-1 Guardar imagen

3. En el cuadro de diálogo **Save Image**, configure los parámetros. Haga clic en **OK** para guardar la imagen.

Figura 2-2 Configuración de parámetros de imagen

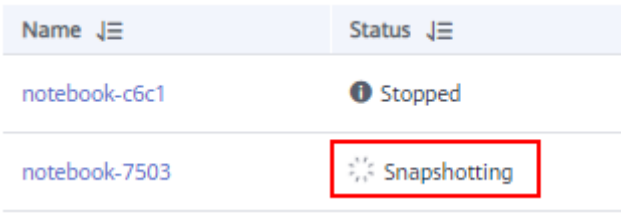


Elija una organización de la lista desplegable **Organization**. Si no hay ninguna organización disponible, haga clic en **Create** a la derecha para crear una.

Los usuarios de una organización pueden compartir todas las imágenes de la organización.

4. La imagen se guardará como una instantánea y tardará unos 5 minutos. Durante este período de tiempo, no realice ninguna operación en la instancia. (Todavía puede realizar operaciones en la página JupyterLab a la que se ha accedido y en el IDE local.)

Figura 2-3 Guardar como instantánea



AVISO

El tiempo necesario para guardar una imagen como instantánea se contará en la duración de ejecución de la instancia. Si la duración de ejecución de la instancia vence antes de guardar la instantánea, se producirá un error al guardar la imagen.

5. Después de guardar la imagen, el estado de la instancia cambia a **Running**. Vea la imagen en la página **Image Management**.

6. Haga clic en el nombre de la imagen para ver sus detalles.

2.2.4 Uso de una imagen personalizada para crear una instancia de notebook

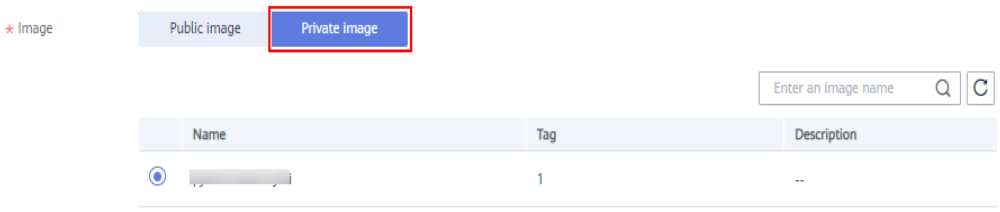
Las imágenes guardadas de una instancia de notebook están disponibles en la página **Image Management**. Estas imágenes se pueden utilizar para crear nuevas instancias de notebook, que heredan las configuraciones de software de las instancias de notebook originales.

Para obtener más información acerca de cómo crear una instancia de notebook con una imagen guardada, consulte la sección **Creación de una instancia de notebook**.

Puede utilizar cualquiera de los siguientes métodos:

Método 1: En la página **Create Notebook**, haga clic en **Private Image** y seleccione la imagen guardada.

Figura 2-4 Selección de una imagen personalizada para crear una instancia de notebook



Método 2: En la página **Image Management Service**, haga clic en la imagen de destino para cambiar a la página de detalles de la imagen. A continuación, haga clic en **Create Notebook**.

3 Uso de una imagen personalizada para entrenar modelos (entrenamiento de nueva versión)

3.1 Descripción general

Los algoritmos suscritos y los marcos incorporados se pueden usar en la mayoría de los escenarios de entrenamiento. En ciertos escenarios de ModelArts le permite crear las imágenes personalizadas para entrenar modelos.

La personalización de una imagen requiere una comprensión profunda de los contenedores. Utilice este método solo si los algoritmos suscritos y los marcos integrados no pueden cumplir con sus requisitos. Las imágenes personalizadas se pueden usar para entrenar modelos de ModelArts solo después de que se suban al Software Repository for Container (SWR).

Puede utilizar imágenes personalizadas para el entrenamiento de ModelArts de cualquiera de las siguientes maneras:

- Uso de una imagen preestablecida con personalización

  Si utiliza una imagen preestablecida para crear un trabajo de entrenamiento y necesita modificar o agregar algunas dependencias de software basadas en la imagen preestablecida, puede personalizar la imagen preestablecida. En este caso, seleccione una imagen preestablecida y elija **Customization** en el cuadro de lista desplegable de la versión del marco.

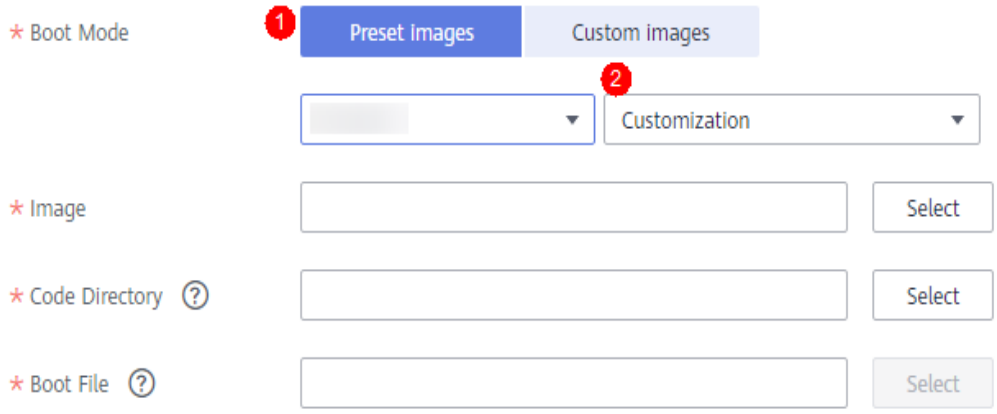
- Uso de una imagen personalizada

  Puede crear una imagen basada en las especificaciones de imagen ModelArts, seleccionar su propia imagen y configurar el directorio de código (opcional) y el comando boot para crear un trabajo de entrenamiento.

Uso de una imagen preestablecida con personalización

Entre este método y crear un trabajo de entrenamiento totalmente basado en una imagen preestablecida, la única diferencia es que debe seleccionar una imagen. Puede crear una imagen personalizada basada en una imagen preestablecida.

Figura 3-1 Crear un algoritmo usando una imagen preestablecida con personalización

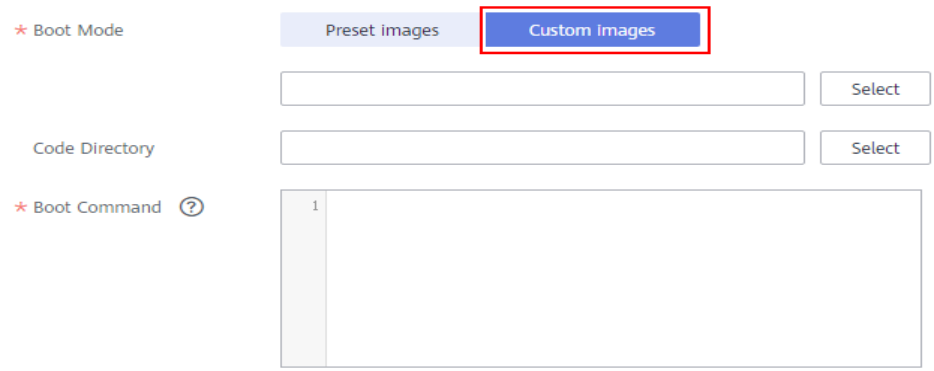


El proceso de este método es el mismo que el de crear un trabajo de entrenamiento basado en una imagen preestablecida. Por ejemplo:

- El sistema inyecta automáticamente variables de entorno.
  - PATH=${PATH}:${MA_HOME}/anaconda/bin
  - LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:${MA_HOME}/anaconda/lib
  - PYTHONPATH=${PYTHONPATH}:${MA_JOB_DIR}
- El archivo de arranque seleccionado se iniciará automáticamente con los comandos de Python. Asegúrese de que el entorno de Python sea correcto. La variable de entorno de PATH se inyecta automáticamente. Ejecute los siguientes comandos para comprobar la versión de Python para el trabajo de entrenamiento:
  - export MA_HOME=/home/ma-user; docker run --rm {image} ${MA_HOME}/anaconda/bin/python -V
  - docker run --rm {image} $(which python) -V
- El sistema agrega automáticamente hiperparámetros asociados con la imagen preestablecida.

Uso de una imagen personalizada

Figura 3-2 Creación de un algoritmo mediante una imagen personalizada



Para obtener más información sobre cómo utilizar imágenes personalizadas compatibles con la nueva versión de entrenamiento, consulte **Uso de una imagen personalizada para entrenar modelos**.

3.2 Preparación de una imagen de entrenamiento

3.2.1 Especificaciones a las Imágenes personalizadas para trabajos de entrenamiento

Cuando utilice un modelo desarrollado localmente y un script de entrenamiento para crear una imagen personalizada, asegúrese de que la imagen personalizada cumple las especificaciones definidas por ModelArts.

NOTA

Tanto en la gestión de entrenamiento de nueva versión como en la de la versión antigua, se pueden utilizar las imágenes personalizadas para crear los trabajos de entrenamiento. Este documento describe la gestión del entrenamiento de la nueva versión. La versión anterior será descontinuada pronto. Se recomienda utilizar la nueva versión.

Especificaciones

- El tamaño de una imagen personalizada no puede exceder los 30 GB. Se recomienda que el tamaño sea inferior o igual a 15 GB. Una imagen sobredimensionada afecta al inicio de un trabajo de entrenamiento.
- El **uid** del usuario predeterminado de una imagen personalizada debe ser **1000**.
- No se puede instalar ningún controlador de GPU o Ascend en una imagen personalizada. Cuando se usa GPU para ejecutar un trabajo de entrenamiento, el ModelArts coloca automáticamente el controlador de GPU en **/usr/local/nvidia** del entorno de entrenamiento. Cuando se usa Ascend para ejecutar un trabajo de entrenamiento, ModelArts coloca automáticamente el controlador Ascend en **/usr/local/Ascend/driver**.
- Las imágenes personalizadas basadas en x86 o Arm pueden ejecutarse solo con las especificaciones correspondientes a su arquitectura.
  - Ejecute el siguiente comando para comprobar la arquitectura de CPU de una imagen personalizada:
    ```
    docker inspect {Custom image path} | grep Architecture
    ```
    A continuación se muestra el resultado del comando para una imagen personalizada basada en Arm:
    ```
    "Architecture": "arm64"
    ```
  - Si el nombre de una especificación contiene **Arm**, esta especificación es una arquitectura de CPU basada en Arm.
  - Si el nombre de una especificación no contiene **Arm**, esta especificación es una arquitectura de CPU basada en x86.

- ModelArts no admite la descarga de paquetes de instalación del código abierto. Instale los paquetes de dependencias requeridos por el trabajo de entrenamiento en la imagen personalizada.

3.2.2 Migración de una imagen a ModelArts

Para migrar una imagen a la nueva versión de gestión de entrenamiento, realice las operaciones siguientes:

1. Agregue el grupo de usuarios predeterminado **ma-group** (GID = 100) para la imagen de la nueva versión de la gestión de entrenamiento. Si existe el grupo de usuarios cuyo GID es 100, omita este paso.
2. Agregue el usuario predeterminado **ma-group** (UID = 1000) para la imagen de la nueva versión de la gestión de entrenamiento. Si existe el usuario cuyo UID es 1000, omita este paso.
3. Modifique el permiso sobre los archivos de la imagen para permitir que **ma-user** cuyo UID es 1000 lea y escriba los archivos.

Puede modificar una imagen haciendo referencia al siguiente Dockerfile para que la imagen cumpla con las especificaciones para imágenes personalizadas de la nueva versión de la gestión de entrenamiento.

```
FROM {An existing image}

USER root

RUN groupadd ma-group -g 100 && \
    useradd -d /home/ma-user -m -u 1000 -g 100 -s /bin/bash ma-user

# [important] avoid missing training log
ENV PYTHONUNBUFFERED=1

USER ma-user
WORKDIR /home/ma-user
```

En la nueva versión de la gestión de entrenamiento, el usuario cuyo UID es 1000 se usa por defecto para ejecutar las imágenes personalizadas. Depure el permiso de los archivos de imagen localmente antes de usar la imagen en la nueva versión de la gestión de entrenamiento. Para depurar el permiso de los archivos de imagen:

Ejecute el siguiente comando para especificar el usuario cuyo UID es 1000 para ejecutar la imagen:

```
docker run -u 1000 -ti {Custom image} bash
```

Ejecute el comando boot (como el siguiente comando) en el contenedor en tiempo de ejecución de imagen:

```
python train.py
```

Compruebe si se produce el error **Permission denied** para comprobar si el permiso de archivo requerido está disponible.

Si se muestra **Permission denied** para un archivo, por ejemplo, **train.py**, ejecute el siguiente comando para cambiar el propietario del archivo para que el archivo pertenezca al usuario cuyo UID es 1000:

```
chown 1000 train.py
```

Sube la imagen adaptada a SWR. Para obtener más información, véase **¿Cómo puedo iniciar sesión en SWR y cargar imágenes en él?**. A continuación, utilice la imagen de ModelArts haciendo referencia a **Uso de una imagen personalizada para crear un trabajo de entrenamiento basado en CPU o GPU**.

3.3 Creación de un algoritmo mediante una imagen personalizada

Sus algoritmos desarrollados localmente o desarrollados utilizando otras herramientas se pueden cargar en ModelArts para una gestión unificada.

Entradas para crear un algoritmo

Puede crear un algoritmo usando una imagen personalizada de ModelArts de cualquiera de las siguientes maneras:

- Entrada 1: En la consola de ModelArts, elija **Algorithm Management** > **My algorithms**. A continuación, cree un algoritmo y utilícelo en los trabajos de entrenamiento o publíquelo en AI Gallery.
- Entrada 2: En la consola de ModelArts, elija **Training Management** > **Training Jobs** y haga clic en **Create Training Job** para crear un algoritmo personalizado y enviar un trabajo de entrenamiento. Para obtener más información, véase **Uso de una imagen personalizada para crear un trabajo de entrenamiento basado en CPU o GPU**.

Parámetros para crear un algoritmo

Figura 3-3 Creación de un algoritmo mediante una imagen personalizada

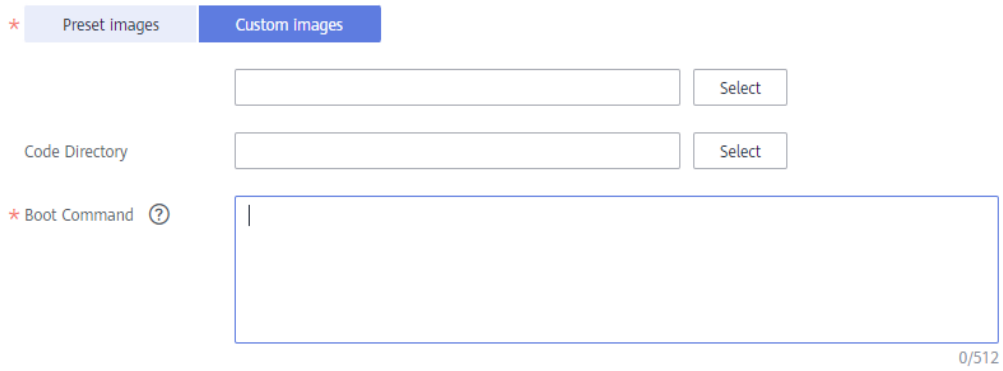


Tabla 3-1 Parámetros para crear un algoritmo

Parámetro	Descripción
Boot Mode	Seleccione **Custom images**. Este parámetro es obligatorio.

Parámetro	Descripción
Image Path	URL de una imagen de SWR. Este parámetro es obligatorio. ● Imágenes privadas o imágenes compartidas: Haga clic en **Select** a la derecha para seleccionar una imagen de SWR. Asegúrese de que la imagen se ha cargado a SWR. ● Imágenes públicas: También puede introducir manualmente la ruta de la imagen en el formato "<Organización a la que pertenece la imagen>/<Nombre de la imagen>" en SWR. No contenga el nombre de dominio (swr.<region>.xxx.com) en la ruta porque el sistema agregará automáticamente el nombre de dominio a la ruta. Por ejemplo: `modelarts-job-dev-image/pytorch_1_8:train-pytorch_1.8.0-cuda_10.2-py_3.7-euleros_2.10.1-x86_64-8.1.1`
Code Directory	Ruta de OBS para almacenar el código de entrenamiento. Este parámetro es opcional. Tome la ruta de OBS **obs://obs-bucket/training-test/demo-code** como ejemplo. El contenido de la ruta OBS se descargará automáticamente a **${MA_JOB_DIR}/demo-code** en el contenedor de entrenamiento, y **demo-code** (personalizable) es el directorio de último nivel de la ruta OBS.
Boot Command	Comando para arrancar una imagen. Este parámetro es obligatorio. El comando de arranque se ejecutará automáticamente después de descargar el directorio de código. ● Si el script de arranque de entrenamiento es un archivo .py, **train.py** por ejemplo, el comando boot puede ser **python ${MA_JOB_DIR}/demo-code/train.py**. ● Si el script de arranque de entrenamiento es un archivo .sh, **main.sh** por ejemplo, el comando boot puede ser **bash ${MA_JOB_DIR}/demo-code/main.sh**. Los puntos y coma (;) y ampersands (&&) se pueden usar para combinar varios comandos de arranque, pero no se admiten saltos de línea. **demo-code** (personalizable) en el comando de arranque es el directorio del último nivel de la ruta OBS.

Configuración de tuberías

Un algoritmo basado en imágenes preestablecidas obtiene datos de un bucket de OBS o conjunto de datos para el entrenamiento del modelo. La salida de entrenamiento se almacena en un bucket de OBS. Los parámetros de entrada y salida del código del algoritmo deben analizarse para permitir el intercambio de datos entre ModelArts y OBS. Para obtener más información sobre cómo desplegar el código para entrenamientos en ModelArts, consulte **Despliegue de un script personalizado**.

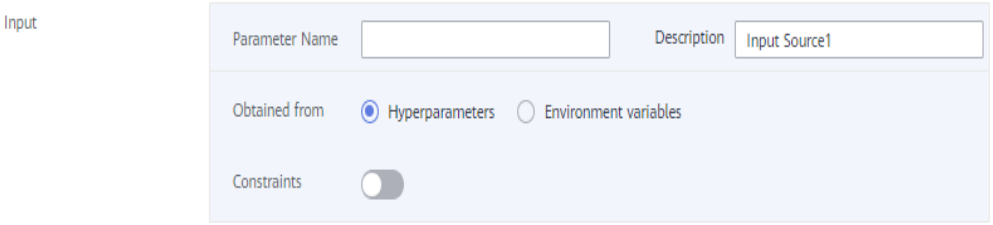
Cuando utilice una imagen preestablecida para crear un algoritmo, configure las canalizaciones de entrada y salida.

- Configuraciones de entrada

Tabla 3-2 Configuraciones de entrada

Parámetro	Descripción
Parameter Name	Si utiliza **argparse** en el código del algoritmo para analizar **data_url** en la entrada de datos, establezca el parámetro de entrada de datos en **data_url** al crear el algoritmo. Establezca el nombre basado en el parámetro de entrada de datos en su código de algoritmo. El parámetro de ruta de código debe ser el mismo que el de entrada de datos analizado en su código de algoritmo. De lo contrario, el código de algoritmo no puede obtener los datos de entrada.
Description	Descripción personalizable del parámetro de entrada,
Obtained from	Fuente del parámetro de entrada. Puede seleccionar **Hyperparameters** (predeterminado) o **Environment variables**.
Constraints	Si los datos se obtienen de una ruta de almacenamiento o de un conjunto de datos de ModelArts. Si selecciona el conjunto de datos de ModelArts como el origen de datos, se agregan las siguientes restricciones: ● **Labeling Type**: Para obtener más información, consulte la sección **Creación de un trabajo de etiquetado**. ● **Data Format** que puede ser **Default**, **CarbonData** o ambos. **Default** indica el formato de manifiesto. ● **Data Segmentation**: solo está disponible para los conjuntos de datos de clasificación de imágenes, de detección de objetos, de clasificación de texto y de clasificación de sonido. Los valores posibles son **Segmented dataset**, **Dataset not segmented** y **Unlimited**. Para obtener más información, consulte la sección **Publicación de una versión de datos**.
Yes	Permite múltiples fuentes de entrada de datos basadas en el algoritmo

Figura 3-4 Configuraciones de entrada

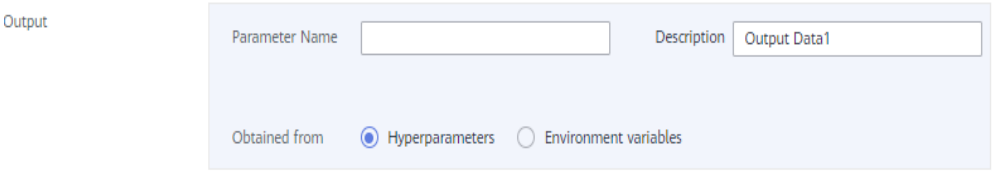


- Configuraciones de salida

Tabla 3-3 Configuraciones de salida

Parámetro	Descripción
Parameter Name	Si utiliza **argparse** en el código del algoritmo para analizar **train_url** en la salida de datos, establezca el parámetro de salida de datos en **train_url** al crear el algoritmo. Establezca el nombre basado en el parámetro de salida de datos en su código de algoritmo. El parámetro de ruta de código debe ser el mismo que el parámetro de salida de datos analizado en el código de algoritmo. De lo contrario, el código de algoritmo no puede obtener la ruta de salida.
Description	Descripción personalizable del parámetro de salida,
Obtained from	Fuente del parámetro de salida. Puede seleccionar **Hyperparameters** (predeterminado) o **Environment variables**.
Yes	Permite múltiples rutas de salida de datos basadas en el algoritmo

Figura 3-5 Configuraciones de salida



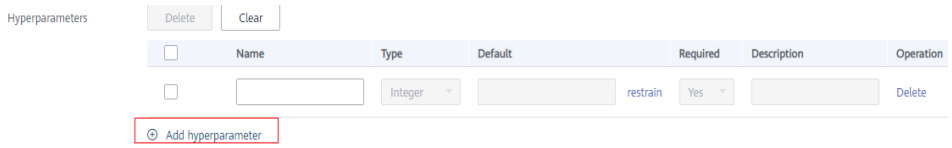
Definición de Hiperparámetros

Cuando se utiliza una imagen preestablecida para crear un algoritmo, ModelArts le permite personalizar los hiperparámetros para que pueda verlos o modificarlos en cualquier momento. Después de definir los hiperparámetros, se muestran en el comando de inicio y se transfieren al archivo de inicio como parámetros de CLI.

1. Importar hiperparámetros.

   Puede hacer clic en **Add hyperparameter** para agregar manualmente hiperparámetros. Solo letras, dígitos, guiones (-), guiones bajos (_), comas (,), puntos (.) y espacios están permitidos.

   Figura 3-6 Adición de hiperparámetros

2. Editar hiperparámetros. Para obtener más información, véase **Tabla 3-4**.

Tabla 3-4 Hiperparámetros

Parámetro	Descripción
Name	Nombre de hiperparámetro. Introduzca de 1 a 64 caracteres. Solo se permiten letras, dígitos, guiones medios (-) y guiones bajos (_).
Type	Tipo del hiperparámetro, que puede ser **String**, **Integer**, **Float**, o **Boolean**
Default	Valor predeterminado del hiperparámetro, que se utiliza para los trabajos de entrenamiento de forma predeterminada
Constraints	Haga clic en **restrain**. A continuación, establezca el rango del valor predeterminado o el valor enumerado en el cuadro de diálogo que se muestra.
Required	Si el parámetro es obligatorio. El valor puede ser **Yes** o **No**. Si selecciona **No**, puede eliminar el hiperparámetro en la página de creación de trabajos de entrenamiento cuando utilice este algoritmo para crear un trabajo de entrenamiento. Si selecciona **Yes**, el hiperparámetro no se puede eliminar.
Description	Descripción del hiperparámetro Solo se permiten letras, dígitos, espacios, guiones (-), guiones bajos (_), comas (,) y puntos (.).

Adición de restricciones de entrenamiento

Puede agregar restricciones de entrenamiento del algoritmo en función de sus necesidades.

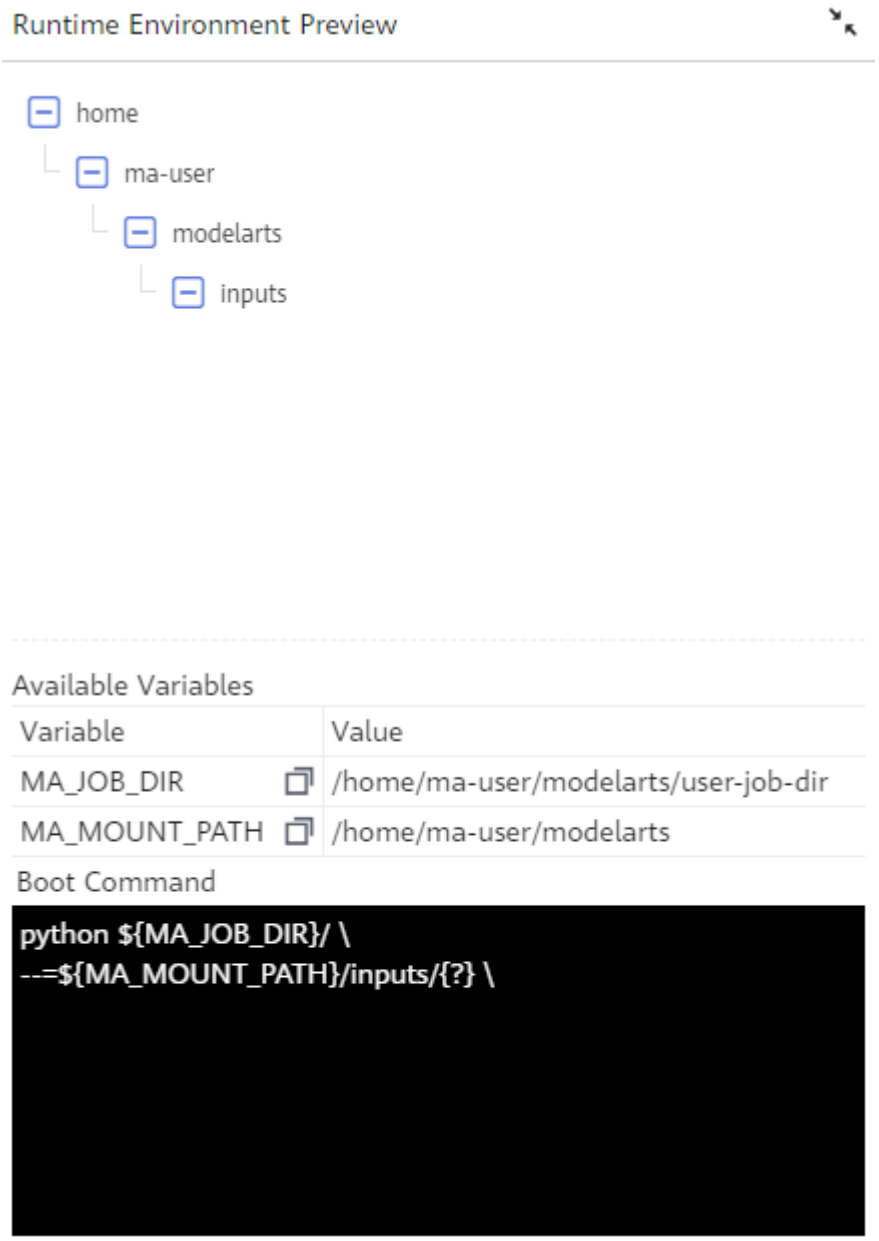
- **Resource Type**: Las opciones son **CPU** y **GPU**. Puede seleccionar varias opciones.
- **Multicard Training**: Seleccione **supported** o **not supported**.
- **Distributed Training**: Seleccione **supported** o **not supported**.

Vista previa del entorno en tiempo de ejecución

Cuando cree un trabajo de entrenamiento, haga clic en la flecha de

en la esquina inferior derecha de la página para conocer la ruta del directorio de código, el archivo de inicio y los datos de entrada y salida en el contenedor de entrenamiento.

Figura 3-7 Vista previa del entorno en tiempo de ejecución



Procedimiento posterior

Después de crear un algoritmo, úsalo para crear un trabajo de entrenamiento. Para obtener más información, véase **Uso de una imagen personalizada para crear un trabajo de entrenamiento basado en CPU o GPU**.

También puede publicar el nuevo algoritmo en AI Gallery y compartirlo con otros usuarios.

3.4 Uso de una imagen personalizada para crear un trabajo de entrenamiento basado en CPU o GPU

El entrenamiento de modelos es un proceso de optimización iterativo. A través de la gestión de entrenamiento unificada, puede seleccionar de forma flexible algoritmos, datos e hiperparámetros para obtener la configuración y el modelo de entrada óptimos. Después de comparar métricas entre versiones de trabajo, puede determinar el trabajo de entrenamiento más satisfactorio.

Prerrequisitos

- Los datos que se van a entrenar se han subido a un directorio de OBS.
- Se ha creado al menos una carpeta vacía para almacenar la salida de entrenamiento en OBS.
- Se ha creado una imagen personalizada basada en las especificaciones de ModelArts. Para obtener más información sobre las especificaciones de imágenes personalizadas, consulte **Especificaciones a las Imágenes personalizadas para trabajos de entrenamiento**.
- La imagen personalizada se ha subido a SWR. Para obtener más información, véase **¿Cómo puedo iniciar sesión en SWR y cargar imágenes en él?**.

Creación de un trabajo de entrenamiento

1. Inicie sesión en la consola de gestión ModelArts. En el panel de navegación izquierdo, elija **Training Management** > **Training Jobs**. Se muestra la lista de trabajos de entrenamiento.
2. Haga clic en **Create Training Job** y configure los parámetros. **Tabla 3-5** enumera los parámetros.

Tabla 3-5 Parámetros de trabajo

Parámetro	Descripción
Created By	Seleccione un algoritmo personalizado. Este parámetro es obligatorio. Si ha creado un algoritmo basado en una imagen personalizada en **Algorithm Management**, elija el algoritmo creado en **My Algorithms**.
Boot Mode	Seleccione **Custom images**. Este parámetro es obligatorio.

Parámetro	Descripción
Image Path	URL de una imagen de SWR. Este parámetro es obligatorio. ● Imágenes privadas o imágenes compartidas: Haga clic en **Select** a la derecha para seleccionar una imagen de SWR. Asegúrese de que la imagen se ha cargado a SWR. ● Imágenes públicas: También puede introducir manualmente la ruta de la imagen en el formato "<Organización a la que pertenece la imagen>/<Nombre de la imagen>" en SWR. No contenga el nombre de dominio (swr.<region>.xxx.com) en la ruta porque el sistema agregará automáticamente el nombre de dominio a la ruta. Por ejemplo: `modelarts-job-dev-image/pytorch_1_8:train-pytorch_1.8.0-cuda_10.2-py_3.7-euleros_2.10.1-x86_64-8.1.1`
Code Directory	Ruta de OBS para almacenar el código de entrenamiento. Este parámetro es opcional. Tome la ruta de OBS **obs://obs-bucket/training-test/demo-code** como ejemplo. El código de entrenamiento en esta ruta se descargará automáticamente a **${MA_JOB_DIR}/demo-code** en el contenedor de entrenamiento, donde **demo-code** es el directorio del último nivel de la ruta de OBS y se puede personalizar.
Boot Command	Comando para arrancar una imagen. Este parámetro es obligatorio. El comando de arranque se ejecutará automáticamente después de descargar el directorio de código. ● Si el script de inicio de entrenamiento es un archivo .py, **train.py** por ejemplo, el comando de arranque puede ser **python ${MA_JOB_DIR}/demo-code/train.py**. ● Si el script de inicio de entrenamiento es un archivo .sh, **main.sh** por ejemplo, el comando de arranque puede ser **bash ${MA_JOB_DIR}/demo-code/main.sh**, donde **demo-code** es el directorio del último nivel de la ruta de OBS y se puede personalizar.

Parámetro	Descripción

Training Input - Parameter Name	El valor recomendado es **data_url**, que debe ser el mismo que el parámetro para analizar los datos de entrada en el código de entrenamiento. Puede establecer varios parámetros de entrada de entrenamiento. El nombre de cada parámetro de entrada de entrenamiento debe ser único, por ejemplo, **car_data_url**, **dog_data_url** y **cat_data_url**.
	Por ejemplo, si utiliza **argparse** en el código de entrenamiento para analizar **data_url** en la entrada de datos, establezca el nombre del parámetro de la entrada del entrenamiento en **data_url**.
	`import argparse`
	`# Create a parsing task.`
	`parser = argparse.ArgumentParser(description="train mnist", formatter_class=argparse.ArgumentDefaultsHelpFormatter)`
	`# Add parameters.`
	`parser.add_argument('--train_url', type=str, help='the path model saved')`
	`parser.add_argument('--data_url', type=str, help='the training data')`
	`# Parse the parameters.`
	`args, unknown = parser.parse_known_args()`

Training Input - Data Path	Seleccione **Dataset** o **Data path** como la entrada de entrenamiento. Si selecciona **Data path**, establezca una ruta de OBS como la entrada de entrenamiento. Cuando comienza el entrenamiento, los datos de la ruta especificada se descargarán automáticamente al contenedor de entrenamiento. Tome la ruta de OBS **obs://obs-bucket/training-test/data** como ejemplo. Los datos se descargarán automáticamente al **${MA_MOUNT_PATH}/inputs/${data_url}_N** del contenedor de entrenamiento. El valor de **N** es el número de parámetros de entrada de entrenamiento menos 1. Por ejemplo: ● Si solo hay un parámetro de entrada de entrenamiento **data_url**, los datos se descargarán automáticamente al **${MA_MOUNT_PATH}/inputs/data_url_0/** del contenedor de entrenamiento. ● Si hay múltiples parámetros de entrada de entrenamiento **car_data_url**, **dog_data_url** y **cat_data_url**, los datos de entrenamiento se descargarán automáticamente a **${MA_MOUNT_PATH}/inputs/car_data_url_0/**, **${MA_MOUNT_PATH}/inputs/dog_data_url_1/** y **${MA_MOUNT_PATH}/inputs/cat_data_url_2/** del contenedor, respectivamente.
Training Output - Parameter Name	El valor recomendado es **train_url**, que debe ser el mismo que el parámetro para analizar los datos de salida en el código de entrenamiento. Puede establecer varios parámetros de salida de entrenamiento. El nombre de cada parámetro de salida de entrenamiento debe ser único.

Parámetro	Descripción
Training Output - Data Path	Seleccione una ruta de OBS como la salida de entrenamiento. Para minimizar los errores, seleccione un directorio vacío. El archivo de resultados de entrenamiento en el contenedor de entrenamiento **${MA_MOUNT_PATH}/outputs/${train_url}_N/** se cargará automáticamente a **obs://obs-bucket/training-test/output**. El valor de **N** es el número de parámetros de salida de entrenamiento menos 1. Por ejemplo: ● Si solo hay un parámetro de salida de entrenamiento **train_url**, el directorio de contenedor de la salida de entrenamiento es **${MA_MOUNT_PATH}/outputs/data_url_0/**. ● Si hay múltiples parámetros de salida de entrenamiento, por ejemplo **car_train_url**, **dog_train_url** y **cat_train_url**, los directorios de contenedores de la salida de entrenamiento son **${MA_MOUNT_PATH}/outputs/car_train_url_0/**, **${MA_MOUNT_PATH}/outputs/dog_train_url_1/** y **${MA_MOUNT_PATH}/outputs/cat_train_url_2/** respectivamente.
Hyperparameters	Se utiliza para entrenar tuning. Este parámetro es opcional.
Environment Variable	Después de iniciar el contenedor, el sistema carga las variables de entorno predeterminadas y las personalizadas aquí. **Tabla 3-6** muestra las variables de entorno predeterminadas.
Auto Restart	Después de activar esta función, puede establecer el número de veces de reinicio para una falla de entrenamiento. Este parámetro es opcional.

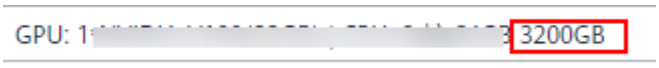
Tabla 3-6 Variables de entorno predeterminadas

Variable de entorno	Descripción
MA_JOB_DIR	Directorio padre del directorio de código.
MA_MOUNT_PATH	Directorio padre de los directorios de entrada y salida de entrenamiento.
VC_TASK_INDEX	Índice de contenedores, a partir de **0**. Este parámetro no tiene sentido para el entrenamiento independiente. En los trabajos de entrenamiento de multinodos, puede utilizar este parámetro para determinar la lógica del algoritmo del contenedor.
MA_NUM_HOSTS	Número de nodos de cómputo, que se obtiene automáticamente de **Compute Nodes**.

Variable de entorno	Descripción
${MA_VJ_NAME}-${MA_TASK_NAME}-N.${MA_VJ_NAME}	Nombre de dominio de comunicación de un nodo. Por ejemplo, el nombre de dominio de comunicación del nodo 0 es **${MA_VJ_NAME}-${MA_TASK_NAME}-0.${MA_VJ_NAME}**. **N** indica el número de los nodos de cómputo. Por ejemplo, si el número de los nodos de cómputo es **4**, las variables de entorno son las siguientes: **${MA_VJ_NAME}-${MA_TASK_NAME}-0.${MA_VJ_NAME}** **${MA_VJ_NAME}-${MA_TASK_NAME}-1.${MA_VJ_NAME}** **${MA_VJ_NAME}-${MA_TASK_NAME}-2.${MA_VJ_NAME}** **${MA_VJ_NAME}-${MA_TASK_NAME}-3.${MA_VJ_NAME}**

3. Seleccione una variante de instancia. El rango de valores de los parámetros de entrenamiento es coherente con las restricciones de las imágenes personalizadas existentes.

Tabla 3-7 Parámetros de recursos

Parámetro	Descripción
Resource Pool	Los grupos de recursos públicos y dedicados están disponibles para que usted seleccione.
Resource Type	Establezca este parámetro en función del tipo de recurso especificado en la imagen.
Instance Flavor	Seleccione una variante de recurso basado en el tipo de recurso. Asegúrese de que la variante seleccionada es la misma que la variante de entrenamiento especificada en la imagen. Por ejemplo, si se selecciona **GPU** en el código de entrenamiento pero se selecciona **CPU** aquí, el entrenamiento puede fallar. Durante el entrenamiento, ModelArts montará SSD NVME en el directorio **/cache**. Puede utilizar este directorio para almacenar archivos temporales. El tamaño del disco de datos varía según el tipo de recurso. Para evitar que la memoria sea insuficiente durante el entrenamiento, compruebe el tamaño del disco de la variante de instancia seleccionada. 
Compute Nodes	Establezca el número de nodos de cálculo. El valor predeterminado es **1**.

Parámetro	Descripción
Persistent Log Saving	Si habilita esta función, seleccione un directorio de OBS vacío como la ruta de log de trabajos para almacenar los archivos de log generados durante la ejecución del trabajo, por ejemplo **obs://obs-bucket/training-test/output/log**. Este parámetro es opcional. **NOTA** Asegúrese de tener permisos de lectura y escritura en el directorio OBS seleccionado.

4. Haga clic en **Submit** para crear el trabajo de entrenamiento.

   Se necesita un período de tiempo para crear un trabajo de entrenamiento.

   Para ver el estado en tiempo real de un trabajo de entrenamiento, vaya a la lista de los trabajos de entrenamiento y haga clic en su nombre. En la página de detalles del trabajo de entrenamiento que se muestra, vea la información básica del trabajo de entrenamiento. Para obtener más información, consulte **Detalles de trabajos de entrenamiento**.

4 Uso de una imagen personalizada para crear aplicaciones de IA para el despliegue de inferencia

4.1 Especificaciones de imágenes personalizadas para crear aplicaciones de IA

Al crear una imagen personalizada utilizando un modelo desarrollado localmente, asegúrese de que la imagen cumple con las especificaciones de ModelArts.

- No se permite ningún código malicioso.
- El tamaño de una imagen personalizada no puede superar los 10 GB.
- **API externos**

  El puerto de servicio externo para una imagen personalizada debe ser **8080**. La API de inferencia debe ser la misma que el URL definida por **apis** en **config.json**. A continuación, se puede acceder directamente a la API de inferencia cuando se inicia la imagen. A continuación se muestra un ejemplo de acceso a una imagen MNIST. La imagen contiene un modelo entrenado usando un conjunto de datos MNIST y puede identificar los dígitos escritos a mano. En este ejemplo, *Listening IP address* es la dirección IP del contenedor.

  – Modelo de solicitud
  ```
  curl -X POST \ http://{Listening IP address}:8080/ \ -F images=@seven.jpg
  ```
  – Ejemplo de respuesta
  ```
  {"mnist_result": 7}
  ```
- **(Opcional) API de prueba de estado**

  Si los servicios no se deben interrumpir durante una actualización sucesiva, la API de comprobación de estado debe configurarse en **config.json** para ModelArts. La API de la prueba de estado devuelve la situación de estado de un servicio cuando se está ejecutando correctamente, o un error cuando se vuelve defectuoso.

AVISO

La API de comprobación de estado debe configurarse para una actualización continua sin problemas.

A continuación se muestra un ejemplo de API de comprobación de estado:

- URI
  ```
  GET /health
  ```
- Ejemplo de solicitud: curl -X GET \ http://*{Listening IP address}*:8080/health
- Ejemplo de respuesta
  ```
  {"health": "true"}
  ```
- Código de estado

  Tabla 4-1 Código de estado

Código de estado	Mensaje	Descripción
200	OK	Solicitud enviada

- **Salida del archivo de log**

  Configure la salida estándar para que los logs se puedan mostrar correctamente.

- **Archivo de arranque de imagen**

  Para implementar un servicio por lotes, establezca el archivo de inicio de una imagen en **/home/run.sh** y use CMD para establecer la ruta de inicio predeterminada. El siguiente es un ejemplo de Dockerfile:

  CMD ["sh", "/home/run.sh"]

- **Dependencias de imagen**

  Para implementar un servicio por lotes, instale los paquetes de dependencias como Python, JRE/JDK y ZIP en la imagen.

- **(Opcional) Actualización continua sin hits**

  Para asegurarse de que los servicios no se interrumpen durante una actualización sucesiva, establezca HTTP **keep-alive** en **200**. Por ejemplo, Gunicorn no es compatible con Keep-alive de forma predeterminada. Para garantizar una actualización continua sin problemas, instale Gevent y configure **--keep-alive 200 -k gevent** en la imagen. La configuración de los parámetros varía en función del marco de servicio. Configure los parámetros según sea necesario.

- **(Opcional) Salida con gracia de un contenedor**

  Para garantizar que los servicios no se interrumpan durante una mejora continua, el sistema debe capturar las señales SIGTERM en el contenedor y esperar 60 segundos antes de salir del contenedor con gracia. Si la duración es inferior a 60 segundos antes de la salida elegante, los servicios pueden interrumpirse durante la actualización continua. Para garantizar un funcionamiento no interrumpido del servicio, el sistema sale del contenedor después de que recibe las señales SIGTERM y procesa todas las solicitudes recibidas. La duración total no es superior a los 90 segundos. A continuación se muestra el ejemplo **run.sh**:

  ```
  #!/bin/bash
  gunicorn_pid=""
  ```



```
handle_sigterm() {
  echo "Received SIGTERM, send SIGTERM to $gunicorn_pid"
  if [ $gunicorn_pid != "" ]; then
      sleep 60
      kill -15 $gunicorn_pid  # Transfer SIGTERM signals to the Gunicorn
process.
      wait $gunicorn_pid           # Wait until the Gunicorn process stops.
  fi
}

trap handle_sigterm TERM
```

4.2 Creación de una imagen personalizada y su uso para crear una aplicación de IA

Si desea utilizar un motor de IA que no es compatible con ModelArts, cree una imagen personalizada para el motor, importe la imagen a ModelArts y use la imagen para crear aplicaciones de IA. Esta sección describe cómo utilizar una imagen personalizada para crear una aplicación de IA e implementarla como un servicio en tiempo real.

El proceso es el siguiente:

1. **Construcción de una imagen localmente**: Crear un paquete de imagen personalizado localmente. Para obtener más información, consulte las **Especificaciones de imágenes personalizadas para crear aplicaciones de AI**.
2. **Verificación de la imagen localmente y su carga a SWR**: Verificar las API de la imagen personalizada y subir la imagen personalizada a SWR.
3. **Uso de la imagen personalizada para crear una aplicación de IA**: Importar la imagen a la gestión de aplicaciones de IA de ModelArts.
4. **Despliegue de la aplicación de IA como servicio en tiempo real**: Desplegar el modelo como un servicio en tiempo real.

Construcción de una imagen localmente

Esta sección utiliza un host de Linux x86_x64 como ejemplo. Puede comprar un ECS de las mismas especificaciones o utilizar un host local existente para crear una imagen personalizada.

1. Instale Docker. Para obtener más información, consulte los **documentos oficiales de Docker**. A continuación se muestra un ejemplo:
   ```
   curl -fsSL get.docker.com -o get-docker.sh
   sh get-docker.sh
   ```
2. Obtenga la imagen base. Se utiliza Ubuntu 18.04 en este ejemplo.
   ```
   docker pull ubuntu:18.04
   ```
3. Cree la carpeta **self-define-images** y edite **Dockerfile** y **test_app.py** en la carpeta de la imagen personalizada. En el código de ejemplo, el código de aplicación se ejecuta en el marco de Flask.

   La estructura de archivos es la siguiente:
   ```
   self-define-images/
       --Dockerfile
       --test_app.py
   ```

   – **Dockerfile**
   ```
   From ubuntu:18.04
   # Configure the HUAWEI CLOUD source and install Python, Python3-PIP, and
   Flask.
   ```



```
RUN cp -a /etc/apt/sources.list /etc/apt/sources.list.bak && \
  sed -i "s@http://.*security.ubuntu.com@http://
repo.huaweicloud.com@g" /etc/apt/sources.list && \
  sed -i "s@http://.*archive.ubuntu.com@http://
repo.huaweicloud.com@g" /etc/apt/sources.list && \
  apt-get update && \
  apt-get install -y python3 python3-pip && \
  pip3 install  --trusted-host https://repo.huaweicloud.com -i https://
repo.huaweicloud.com/repository/pypi/simple  Flask

# Copy the application code to the image.
COPY test_app.py /opt/test_app.py

# Specify the boot command of the image.
CMD python3  /opt/test_app.py
```

- **test_app.py**

```
from flask import Flask, request
import json
app = Flask(__name__)

@app.route('/greet', methods=['POST'])
def say_hello_func():
    print("----------- in hello func ----------")
    data = json.loads(request.get_data(as_text=True))
    print(data)
    username = data['name']
    rsp_msg = 'Hello, {}!'.format(username)
    return json.dumps({"response":rsp_msg}, indent=4)

@app.route('/goodbye', methods=['GET'])
def say_goodbye_func():
    print("----------- in goodbye func ----------")
    return '\nGoodbye!\n'


@app.route('/', methods=['POST'])
def default_func():
    print("----------- in default func ----------")
    data = json.loads(request.get_data(as_text=True))
    return '\n called default func !\n {} \n'.format(str(data))

# host must be "0.0.0.0", port must be 8080
if __name__ == '__main__':
    app.run(host="0.0.0.0", port=8080)
```

NOTA

ModelArts reenvía las solicitudes al puerto 8080 del servicio iniciado desde la imagen personalizada. Por lo tanto, el puerto de escucha de servicio en el contenedor debe ser el puerto 8080. Consulte el archivo **test_app.py**.

4. Cambie a la carpeta **self-define-images** y ejecute el siguiente comando para crear **test:v1** de imágenes personalizadas:

```
docker build -t test:v1 .
```

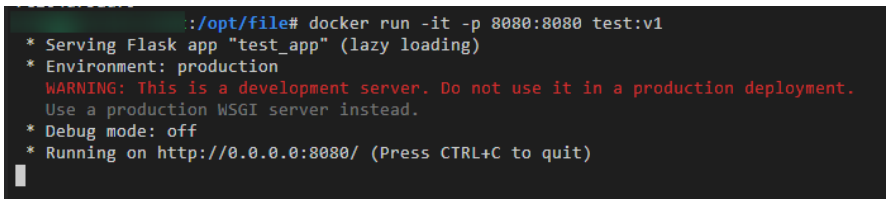
5. Ejecute **docker image** para ver la imagen personalizada que ha creado.

Verificación de la imagen localmente y su carga a SWR

1. Ejecute el siguiente comando en el entorno local para iniciar la imagen personalizada:

```
docker run -it -p 8080:8080 test:v1
```


Figura 4-1 Inicio de una imagen personalizada

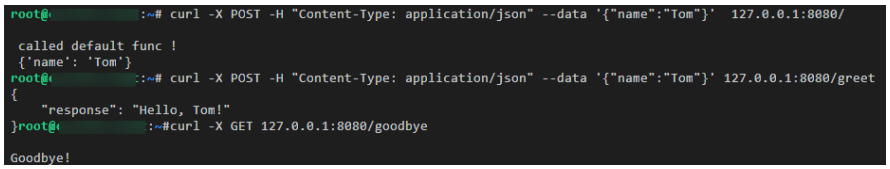


2. Abra otro terminal y ejecute los siguientes comandos para probar las funciones de las tres API de la imagen personalizada:

```
curl -X POST -H "Content-Type: application/json" --data '{"name":"Tom"}'
127.0.0.1:8080/
curl -X POST -H "Content-Type: application/json" --data '{"name":"Tom"}'
127.0.0.1:8080/greet
curl -X GET 127.0.0.1:8080/goodbye
```

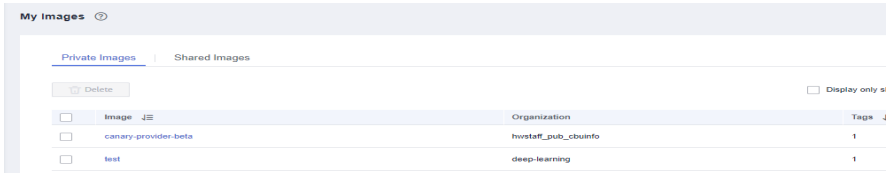
Si se muestra la información similar a la siguiente, la verificación de la función se realiza correctamente.

Figura 4-2 Prueba de las funciones de API



3. Suba la imagen personalizada a SWR. Para obtener más información, consulte **¿Cómo puedo subir imágenes a SWR?**
4. Vea la imagen cargada en la página **My Images** > **Private Images** de la consola de SWR.

Figura 4-3 Imágenes cargadas

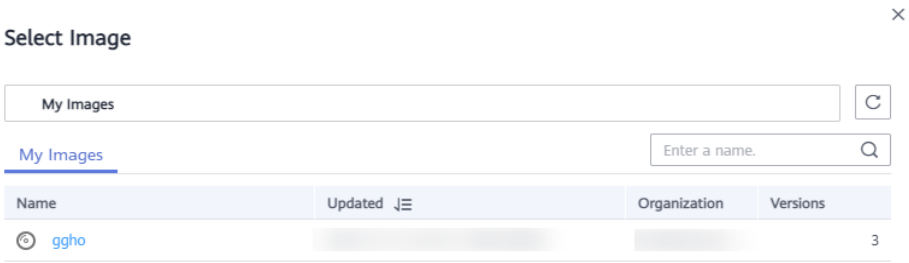


Uso de la imagen personalizada para crear una aplicación de IA

Importe un metamodelo. Para obtener más información, consulte la sección **Creación e importación de una imagen de modelo**. Los parámetros clave son los siguientes:

- **Meta Model Source**: Seleccione **Container image**.
  - **Container Image Path**: Seleccione la imagen privada creada.

Figura 4-4 Imagen privada creada



  - **Container API**: Protocolo y número de puerto para iniciar un modelo. Este parámetro es opcional.
  - **Health Check**: Comprueba el estado de salud de un modelo. Este parámetro es opcional. Este parámetro es configurable solo cuando la API de comprobación de estado está configurada en la imagen personalizada. De lo contrario, la creación de la aplicación AI fallará.

- **APIs**: Las API de una imagen personalizada. Este parámetro es opcional. Las API de modelo deben cumplir con las especificaciones de ModelArts. Para obtener más información, consulte la sección **Especificaciones para compilar el archivo de configuración del modelo**.

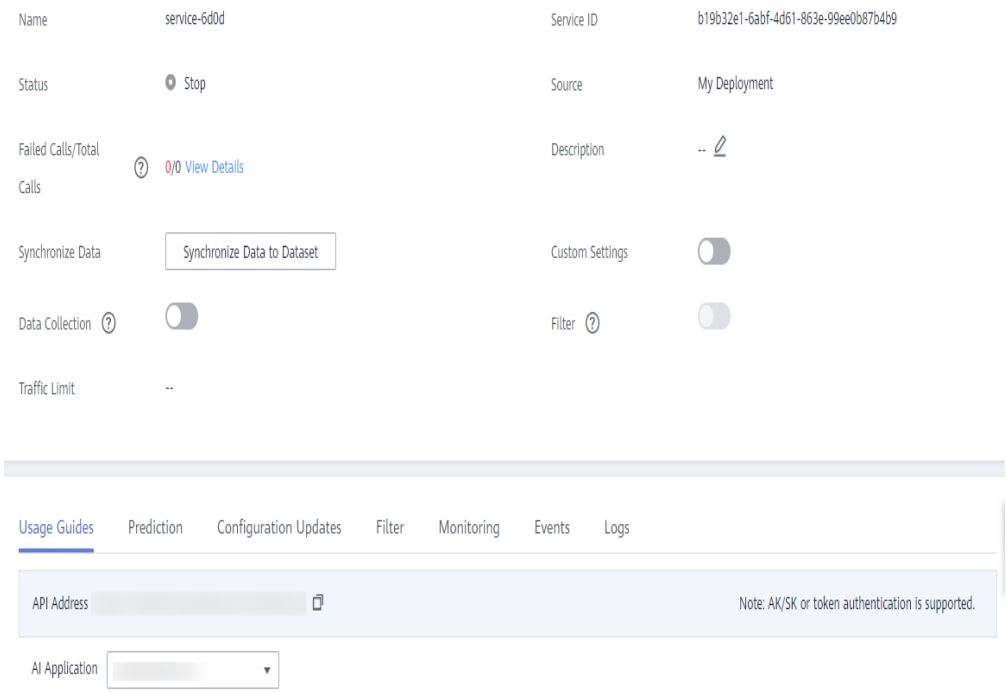
  El archivo de configuración es el siguiente:

```
[{
        "url": "/",
        "method": "post",
        "request": {
            "Content-type": "application/json"
        },
        "response": {
            "Content-type": "application/json"
        }
    },
{
        "url": "/greet",
        "method": "post",
        "request": {
            "Content-type": "application/json"
        },
        "response": {
            "Content-type": "application/json"
        }
    },
{
        "url": "/goodbye",
        "method": "get",
        "request": {
            "Content-type": "application/json"
        },
        "response": {
            "Content-type": "application/json"
        }
    }
]
```

Despliegue de la aplicación de IA como servicio en tiempo real

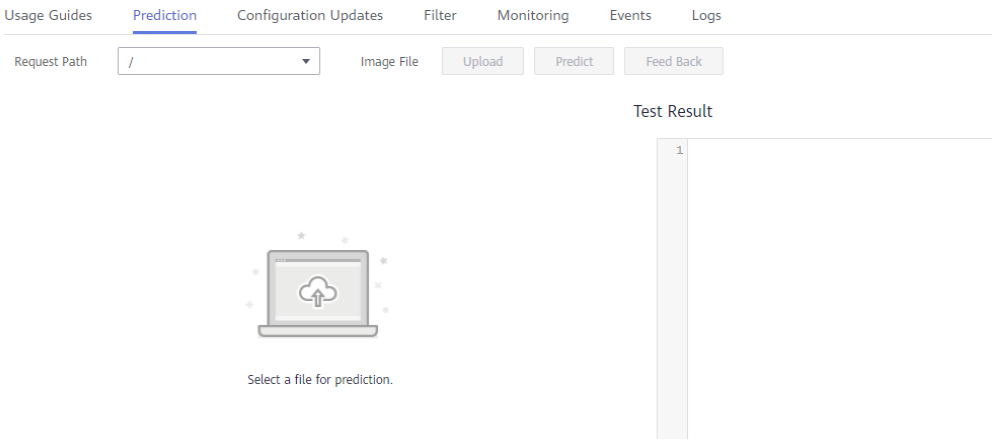
1. Implemente la aplicación de IA como un servicio en tiempo real. Para obtener más información, consulte la sección **Despliegue como servicio en tiempo real**.
2. Vea los detalles sobre el servicio en tiempo real.

Figura 4-5 Guías de uso



3. Acceda al servicio en tiempo real en la página de ficha **Prediction**.

Figura 4-6 Acceso a un servicio en tiempo real



4.3 Imágenes de base de inferencia

4.3.1 Imágenes de base de inferencia disponibles

La inferencia de ModelArts proporciona una serie de imágenes base. Puede crear las imágenes personalizadas basadas en estas imágenes base para implementar los servicios de inferencia.

x86 (CPU/GPU)

Tabla 4-2 TensorFlow

Versión del motor AI	Entorno de tiempo de ejecución	URI
2.1.0	CPU GPU (CUDA 10.1)	swr.{region_id}.myhuaweicloud.com/atelier/tensorflow_2_1:tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64-20220713110657-e769956
1.15.5	CPU GPU (CUDA 11.4)	swr.{region_id}.myhuaweicloud.com/aip/tensorflow_1_15:tensorflow_1.15.5-cuda_11.4-py_3.8-ubuntu_20.04-x86_64-20220524162601-50d6a18
2.6.0	CPU GPU (CUDA 11.2)	swr.{region_id}.myhuaweicloud.com/aip/tensorflow_2_6:tensorflow_2.6.0-cuda_11.2-py_3.7-ubuntu_18.04-x86_64-20220524162601-50d6a18

Tabla 4-3 PyTorch

Versión del motor AI	Entorno de tiempo de ejecución	URI
1.8.0	CPU GPU (CUDA 10.2)	swr.{region_id}.myhuaweicloud.com/atelier/pytorch_1_8:pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64-20220713110657-e769956
1.8.2	CPU GPU (CUDA 11.1)	swr.{region_id}.myhuaweicloud.com/aip/pytorch_1_8:pytorch_1.8.2-cuda_11.1-py_3.7-ubuntu_18.04-x86_64-20220524162601-50d6a18

Tabla 4-4 MindSpore

Versión del motor AI	Entorno de tiempo de ejecución	URI
1.7.0	CPU	swr.{region_id}.myhuaweicloud.com/atelier/mindspore_1_7_0:mindspore_1.7.0-cpu-py_3.7-ubuntu_18.04-x86_64-20220702120711-8590b76
1.7.0	GPU (CUDA 10.1)	swr.{region_id}.myhuaweicloud.com/atelier/mindspore_1_7_0:mindspore_1.7.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64-20220702120711-8590b76

Versión del motor AI	Entorno de tiempo de ejecución	URI
1.7.0	GPU (CUDA 11.1)	swr.{region_id}.myhuaweicloud.com/atelier/mindspore_1_7_0:mindspore_1.7.0-cuda_11.1-py_3.7-ubuntu_18.04-x86_64-20220702120711-8590b76

4.3.2 Imágenes base de inferencia con TensorFlow (CPU/GPU)

ModelArts ofrece las siguientes imágenes de base de inferencia con TensorFlow (CPU/GPU):

- **Versión del motor 1: tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64**
- **Versión del motor 2: tensorflow_1.15.5-cuda_11.4-py_3.8-ubuntu_20.04-x86_64**
- **Versión del motor 3: tensorflow_2.6.0-cuda_11.2-py_3.7-ubuntu_18.04-x86_64**

Versión del motor 1: tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64

- Ruta de la imágen: swr.{region}.myhuaweicloud.com/atelier/tensorflow_2_1:tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64-20220713110657-e769956
- Tiempo de creación de la imagen: 20220713110657 (aaaa-mm-dd-hh-mm-ss)
- Versión del sistema de la imagen: Ubuntu 18.04.4 LTS
- CUDA: 10.1.243
- cuDNN: 7.6.5.32
- Ruta y versión del intérprete de Python: /home/ma-user/anaconda3/envs/TensorFlow-2.1/bin/python, python 3.7.10
- Ruta de instalación de paquetes de terceros: /home/ma-user/anaconda3/envs/TensorFlow-2.1/lib/python3.7/site-packages
- Ciertos paquetes de instalación de pip:

```
Cython                                   0.29.21
easydict                                 1.9
Flask                                    2.0.1
grpcio                                   1.47.0
gunicorn                                 20.1.0
h5py                                     3.7.0
ipykernel                                6.7.0
Jinja2                                   3.0.1
lxml                                     4.9.1
matplotlib                               3.5.1
moxing-framework                         2.1.0.5d9c87c8
numpy                                    1.19.5
opencv-python                            4.1.2.30
pandas                                   1.1.5
Pillow                                   9.2.0
pip                                      22.1.2
protobuf                                 3.20.1
psutil                                   5.8.0
PyYAML                                   5.1
requests                                 2.27.1
scikit-learn                             0.22.1
scipy                                    1.5.2
sklearn                                  0.0
tensorboard                              2.1.1
tensorboardX                             2.0
```



```
tensorflow                              2.1.0
tensorflow-estimator                    2.1.0
wheel                                   0.37.1
zipp                                    3.8.0
...
```

- Ciertos paquetes de instalación de apt:

```
apt
ca-certificates
cmake
cuda
curl
ethtool
fdisk
ffmpeg
g++
gcc
git
gpg
graphviz
libsm6
libxext6
libopencv-dev
libxrender-dev
libatlas3-base
libnuma-dev
libcap-dev
libssl-dev
liblz-dev
libbz2-dev
liblzma-dev
libboost-graph-dev
libsndfile1
libcurl4-openssl-dev
libopenblas-base
liblapack3
libopenblas-dev
libprotobuf-dev
libleveldb-dev
libsnappy-dev
libhdf5-serial-dev
liblapacke-dev
libgflags-dev
libgoogle-glog-dev
liblmdb-dev
libatlas-base-dev
librdmacm1
libcap2-bin
libpq-dev
mysql-common
net-tools
nginx
openslide-tools
openssh-client
openssh-server
openssh-sftp-server
openssl
protobuf-compiler
redis-server
redis-tools
rpm
tar
tofrodos
unzip
vim
wget
zip
zlib1g-dev
...
```


Versión del motor 2: tensorflow_1.15.5-cuda_11.4-py_3.8-ubuntu_20.04-x86_64

- Ruta de la imagen: swr.{region}.myhuaweicloud.com/aip/tensorflow_1_15:tensorflow_1.15.5-cuda_11.4-py_3.8-ubuntu_20.04-x86_64-20220524162601-50d6a18
- Tiempo de creación de la imagen: 20220524162601 (aaaa-mm-dd-hh-mm-ss)
- Versión del sistema de la imagen: Ubuntu 20.04.4 LTS
- CUDA: 11.4.3
- cuDNN: 8.2.4.15
- Ruta y versión del intérprete de Python: /home/ma-user/anaconda3/envs/TensorFlow-1.15.5/bin/python, python 3.8.13
- Ruta de instalación de paquetes de terceros: /home/ma-user/anaconda3/envs/TensorFlow-1.15.5/lib/python3.8/site-packages
- Ciertos paquetes de instalación de pip:

```
Cython 0.29.21
psutil 5.9.0
matplotlib 3.5.1
protobuf 3.20.1
tensorflow 1.15.5+nv
Flask 2.0.1
grpcio 1.46.1
gunicorn 20.1.0
Pillow 9.0.1
tensorboard 1.15.0
PyYAML 6.0
pip 22.0.4
lxml 4.7.1
numpy 1.18.5
tensorflow-estimator 1.15.1
...
```

- Ciertos paquetes de instalación de apt:

```
apt
ca-certificates
cmake
cuda
curl
ethtool
fdisk
ffmpeg
g++
gcc
git
gpg
graphviz
libsm6
libxext6
libopencv-dev
libxrender-dev
libatlas3-base
libnuma-dev
libcap-dev
libssl-dev
liblz-dev
libbz2-dev
liblzma-dev
libboost-graph-dev
libsndfile1
libcurl4-openssl-dev
libopenblas-base
liblapack3
libopenblas-dev
libprotobuf-dev
```



```
libleveldb-dev
libsnappy-dev
libhdf5-serial-dev
liblapacke-dev
libgflags-dev
libgoogle-glog-dev
liblmdb-dev
libatlas-base-dev
librdmacm1
libcap2-bin
libpq-dev
mysql-common
net-tools
nginx
openslide-tools
openssh-client
openssh-server
openssh-sftp-server
openssl
protobuf-compiler
redis-server
redis-tools
rpm
tar
tofrodos
unzip
vim
wget
zip
zlib1g-dev
...
```

Versión del motor 3: tensorflow_2.6.0-cuda_11.2-py_3.7-ubuntu_18.04-x86_64

- Ruta de la imagen: swr.{region}.myhuaweicloud.com/aip/tensorflow_2_6:tensorflow_2.6.0-cuda_11.2-py_3.7-ubuntu_18.04-x86_64-20220524162601-50d6a18
- Tiempo de creación de la imagen: 20220524162601 (aaaa-mm-dd-hh-mm-ss)
- Versión del sistema de la imagen: Ubuntu 18.04.4 LTS
- CUDA: 11.2.0
- cuDNN: 8.1.1.33
- Ruta y versión del intérprete de Python: /home/ma-user/anaconda3/envs/TensorFlow-2.6.0/bin/python, python 3.7.10
- Ruta de instalación de paquetes de terceros: /home/ma-user/anaconda3/envs/TensorFlow-2.6.0/lib/python3.7/site-packages
- Ciertos paquetes de instalación de pip:

```
Cython 0.29.21
requests 2.27.1
easydict 1.9
tensorboardX 2.0
tensorflow 2.6.0
Flask 2.0.1
grpcio 1.46.1
gunicorn 20.1.0
idna 3.3
tensorflow-estimator 2.9.0
pandas 1.1.5
Pillow 9.0.1
lxml 4.8.0
matplotlib 3.5.1
scikit-learn 0.22.1
psutil 5.8.0
PyYAML 5.1
```



```
numpy 1.17.0
opencv-python 4.1.2.30
protobuf 3.20.1
pip 21.2.2
...
```

- Ciertos paquetes de instalación de apt:

```
apt
ca-certificates
cmake
cuda
curl
ethtool
fdisk
ffmpeg
g++
gcc
git
gpg
graphviz
libsm6
libxext6
libopencv-dev
libxrender-dev
libatlas3-base
libnuma-dev
libcap-dev
libssl-dev
liblz-dev
libbz2-dev
liblzma-dev
libboost-graph-dev
libsndfile1
libcurl4-openssl-dev
libopenblas-base
liblapack3
libopenblas-dev
libprotobuf-dev
libleveldb-dev
libsnappy-dev
libhdf5-serial-dev
liblapacke-dev
libgflags-dev
libgoogle-glog-dev
liblmdb-dev
libatlas-base-dev
librdmacm1
libcap2-bin
libpq-dev
mysql-common
net-tools
nginx
openslide-tools
openssh-client
openssh-server
openssh-sftp-server
openssl
protobuf-compiler
redis-server
redis-tools
rpm
tar
tofrodos
unzip
vim
wget
zip
zlib1g-dev
...
```


4.3.3 Imágenes base de inferencia con PyTorch (CPU/GPU)

ModelArts ofrece las siguientes imágenes de base de inferencia con PyTorch (CPU/GPU):

- **Versión del motor 1: pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64**
- **Versión del motor 2: pytorch_1.8.2-cuda_11.1-py_3.7-ubuntu_18.04-x86_64**

Versión del motor 1: pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64

- Ruta de la imagen: swr.{region_id}.myhuaweicloud.com/atelier/pytorch_1_8:pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64-20220713110657-e769956
- Tiempo de creación de la imagen: 20220713110657 (aaaa-mm-dd-hh-mm-ss)
- Versión del sistema de la imagen: Ubuntu 18.04.4 LTS
- CUDA: 10.2.89
- cuDNN: 7.6.5.32
- Ruta y versión del intérprete de Python: /home/ma-user/anaconda3/envs/PyTorch-1.8/bin/python, python 3.7.10
- Ruta de instalación de paquetes de terceros: /home/ma-user/anaconda3/envs/PyTorch-1.8/lib/python3.7/site-packages
- Ciertos paquetes de instalación de pip:

```
Cython                          0.27.3
easydict                        1.9
Flask                           2.0.1
fonttools                       4.34.4
gunicorn                        20.1.0
ipykernel                       6.7.0
Jinja2                          3.0.1
lxml                            4.9.1
matplotlib                      3.5.1
mmcv                            1.2.7
moxing-framework                2.1.0.5d9c87c8
numpy                           1.19.5
opencv-python                   4.1.2.30
pandas                          1.1.5
Pillow                          9.2.0
pip                             22.1.2
protobuf                        3.20.1
psutil                          5.8.0
PyYAML                          5.1
requests                        2.27.1
scikit-learn                    0.22.1
scipy                           1.5.2
sklearn                         0.0
tensorboard                     2.1.1
tensorboardX                    2.0
torch                           1.8.0
torchtext                       0.5.0
torchvision                     0.9.0
tornado                         6.2
tqdm                            4.64.0
traitlets                       5.3.0
typing_extensions               4.3.0
urllib3                         1.26.10
watchdog                        2.0.0
wcwidth                         0.2.5
Werkzeug                        2.1.2
wheel                           0.37.1
yapf                            0.32.0
zipp                            3.8.0
...
```

- Ciertos paquetes de instalación de apt:

```
apt
ca-certificates
cmake
cuda
curl
ethtool
fdisk
ffmpeg
g++
gcc
git
gpg
graphviz
libsm6
libxext6
libopencv-dev
libxrender-dev
libatlas3-base
libnuma-dev
libcap-dev
libssl-dev
liblz-dev
libbz2-dev
liblzma-dev
libboost-graph-dev
libsndfile1
libcurl4-openssl-dev
libopenblas-base
liblapack3
libopenblas-dev
libprotobuf-dev
libleveldb-dev
libsnappy-dev
libhdf5-serial-dev
liblapacke-dev
libgflags-dev
libgoogle-glog-dev
liblmdb-dev
libatlas-base-dev
librdmacm1
libcap2-bin
libpq-dev
mysql-common
net-tools
nginx
openslide-tools
openssh-client
openssh-server
openssh-sftp-server
openssl
protobuf-compiler
redis-server
redis-tools
rpm
tar
tofrodos
unzip
vim
wget
zip
zlib1g-dev
...
```

Versión del motor 2: pytorch_1.8.2-cuda_11.1-py_3.7-ubuntu_18.04-x86_64

- Ruta de la imagen: swr.{region}.myhuaweicloud.com/aip/pytorch_1_8:pytorch_1.8.2-cuda_11.1-py_3.7-ubuntu_18.04-x86_64-20220524162601-50d6a18

- Tiempo de creación de la imagen: 20220524162601 (aaaa-mm-dd-hh-mm-ss)
- Versión del sistema de la imagen: Ubuntu 18.04.4 LTS
- CUDA: 11.1.1
- cuDNN: 8.0.5.39
- Ruta y versión del intérprete de Python: /home/ma-user/anaconda3/envs/PyTorch-1.8.2/bin/python, python 3.7.10
- Ruta de instalación de paquetes de terceros: /home/ma-user/anaconda3/envs/PyTorch-1.8.2/lib/python3.7/site-packages
- Ciertos paquetes de instalación de pip:

```
Cython 0.27.3
mmcv 1.2.7
easydict 1.9
tensorboardX 2.0
torch 1.8.2+cu111
Flask 2.0.1
pandas 1.1.5
gunicorn 20.1.0
PyYAML 5.1
torchaudio 0.8.2
Pillow 9.0.1
psutil 5.8.0
lxml 4.8.0
matplotlib 3.5.1
torchvision 0.9.2+cu111
pip 21.2.2
protobuf 3.20.1
numpy 1.17.0
opencv-python 4.1.2.30
scikit-learn 0.22.1
...
```

- Ciertos paquetes de instalación de apt:

```
apt
ca-certificates
cmake
cuda
curl
ethtool
fdisk
ffmpeg
g++
gcc
git
gpg
graphviz
libsm6
libxext6
libopencv-dev
libxrender-dev
libatlas3-base
libnuma-dev
libcap-dev
libssl-dev
liblz-dev
libbz2-dev
liblzma-dev
libboost-graph-dev
libsndfile1
libcurl4-openssl-dev
libopenblas-base
liblapack3
libopenblas-dev
libprotobuf-dev
libleveldb-dev
libsnappy-dev
```



```
libhdf5-serial-dev
liblapacke-dev
libgflags-dev
libgoogle-glog-dev
liblmdb-dev
libatlas-base-dev
librdmacm1
libcap2-bin
libpq-dev
mysql-common
net-tools
nginx
openslide-tools
openssh-client
openssh-server
openssh-sftp-server
openssl
protobuf-compiler
redis-server
redis-tools
rpm
tar
tofrodos
unzip
vim
wget
zip
zlib1g-dev
...
```

4.3.4 Imágenes base de inferencia con MindSpore (CPU/GPU)

ModelArts ofrece las siguientes imágenes de base de inferencia con MindSpore (CPU/GPU):

- **Versión del motor 1: mindspore_1.7.0-cpu-py_3.7-ubuntu_18.04-x86_64**
- **Versión del motor 2: mindspore_1.7.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64**
- **Versión del motor 3: mindspore_1.7.0-cuda_11.1-py_3.7-ubuntu_18.04-x86_64**

Versión del motor 1: mindspore_1.7.0-cpu-py_3.7-ubuntu_18.04-x86_64

- Ruta de la imagen: swr.{region}.myhuaweicloud.com/atelier/mindspore_1_7_0:mindspore_1.7.0-cpu-py_3.7-ubuntu_18.04-x86_64-20220702120711-8590b76
- Tiempo de creación de la imagen: 20220702120711 (aaaa-mm-dd-hh-mm-ss)
- Versión del sistema de la imagen: Ubuntu 18.04.4 LTS
- Ruta y versión del intérprete de Python: /home/ma-user/anaconda3/envs/MindSpore/bin/python, python 3.7.10
- Ruta de instalación de paquetes de terceros: /home/ma-user/anaconda3/envs/MindSpore/lib/python3.7/site-packages
- Ciertos paquetes de instalación de pip:

```
cycler                        0.11.0
easydict                      1.9
Flask                         2.0.1
grpcio                        1.47.0
gunicorn                      20.1.0
ipykernel                     6.7.0
Jinja2                        3.0.1
lxml                          4.9.0
matplotlib                    3.5.1
mindinsight                   1.7.0
mindspore                     1.7.0
```



```
mindvision                          0.1.0
moxing-framework                    2.1.0.5d9c87c8
numpy                               1.17.0
opencv-contrib-python-headless      4.6.0.66
opencv-python-headless              4.6.0.66
pandas                              1.1.5
Pillow                              9.1.1
pip                                 22.1.2
protobuf                            3.20.1
psutil                              5.8.0
PyYAML                              5.1
requests                            2.27.1
scikit-learn                        0.22.1
scipy                               1.5.2
setuptools                          62.6.0
sklearn                             0.0
tensorboardX                        2.0
threadpoolctl                       3.1.0
tomli                               2.0.1
tornado                             6.1
tqdm                                4.64.0
traitlets                           5.3.0
treelib                             1.6.1
urllib3                             1.26.9
wheel                               0.37.1
zipp                                3.8.0
...
```

- Ciertos paquetes de instalación de apt:

```
apt
ca-certificates
cmake
curl
ethtool
fdisk
ffmpeg
g++
gcc
git
gpg
graphviz
libsm6
libxext6
libopencv-dev
libxrender-dev
libatlas3-base
libnuma-dev
libcap-dev
libssl-dev
liblz-dev
libbz2-dev
liblzma-dev
libboost-graph-dev
libsndfile1
libcurl4-openssl-dev
libopenblas-base
liblapack3
libopenblas-dev
libprotobuf-dev
libleveldb-dev
libsnappy-dev
libhdf5-serial-dev
liblapacke-dev
libgflags-dev
libgoogle-glog-dev
liblmdb-dev
libatlas-base-dev
librdmacm1
libcap2-bin
libpq-dev
```



```
mysql-common
net-tools
nginx
openslide-tools
openssh-client
openssh-server
openssh-sftp-server
openssl
protobuf-compiler
redis-server
redis-tools
rpm
tar
tofrodos
unzip
vim
wget
zip
zlib1g-dev
...
```

Versión del motor 2: mindspore_1.7.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64

- Ruta de la imagen: swr.{region}.myhuaweicloud.com/atelier/mindspore_1_7_0:mindspore_1.7.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64-20220702120711-8590b76
- Tiempo de creación de la imagen: 20220702120711 (aaaa-mm-dd-hh-mm-ss)
- Versión del sistema de la imagen: Ubuntu 18.04.4 LTS
- CUDA: 10.1.243
- cuDNN: 7.6.5.32
- Ruta y versión del intérprete de Python: /home/ma-user/anaconda3/envs/MindSpore/bin/python, python 3.7.10
- Ruta de instalación de paquetes de terceros: /home/ma-user/anaconda3/envs/MindSpore/lib/python3.7/site-packages
- Ciertos paquetes de instalación de pip:

```
cycler                          0.11.0
easydict                        1.9
Flask                           2.0.1
grpcio                          1.47.0
gunicorn                        20.1.0
ipykernel                       6.7.0
Jinja2                          3.0.1
lxml                            4.9.0
matplotlib                      3.5.1
mindinsight                     1.7.0
mindspore                       1.7.0
mindvision                      0.1.0
moxing-framework                2.1.0.5d9c87c8
numpy                           1.17.0
opencv-contrib-python-headless  4.6.0.66
opencv-python-headless          4.6.0.66
pandas                          1.1.5
Pillow                          9.1.1
pip                             22.1.2
protobuf                        3.20.1
psutil                          5.8.0
PyYAML                          5.1
requests                        2.27.1
scikit-learn                    0.22.1
scipy                           1.5.2
setuptools                      62.6.0
sklearn                         0.0
tensorboardX                    2.0
```



```
threadpoolctl                   3.1.0
tomli                           2.0.1
tornado                         6.1
tqdm                            4.64.0
traitlets                       5.3.0
treelib                         1.6.1
urllib3                         1.26.9
wheel                           0.37.1
zipp                            3.8.0
...
```

- Ciertos paquetes de instalación de apt:

```
apt
ca-certificates
cmake
cuda
curl
ethtool
fdisk
ffmpeg
g++
gcc
git
gpg
graphviz
libsm6
libxext6
libopencv-dev
libxrender-dev
libatlas3-base
libnuma-dev
libcap-dev
libssl-dev
liblz-dev
libbz2-dev
liblzma-dev
libboost-graph-dev
libsndfile1
libcurl4-openssl-dev
libopenblas-base
liblapack3
libopenblas-dev
libprotobuf-dev
libleveldb-dev
libsnappy-dev
libhdf5-serial-dev
liblapacke-dev
libgflags-dev
libgoogle-glog-dev
liblmdb-dev
libatlas-base-dev
librdmacm1
libcap2-bin
libpq-dev
mysql-common
net-tools
nginx
openslide-tools
openssh-client
openssh-server
openssh-sftp-server
openssl
protobuf-compiler
redis-server
redis-tools
rpm
tar
tofrodos
unzip
vim
```



```
wget
zip
zlib1g-dev
...
```

Versión del motor 3: mindspore_1.7.0-cuda_11.1-py_3.7-ubuntu_18.04-x86_64

- Ruta de la imagen: swr.{region}.myhuaweicloud.com/atelier/mindspore_1_7_0:mindspore_1.7.0-cuda_11.1-py_3.7-ubuntu_18.04-x86_64-20220702120711-8590b76
- Tiempo de creación de la imagen: 20220702120711 (aaaa-mm-dd-hh-mm-ss)
- Versión del sistema de la imagen: Ubuntu 18.04.4 LTS
- CUDA: 11.1.1
- cuDNN: 8.0.5.39
- Ruta y versión del intérprete de Python: /home/ma-user/anaconda3/envs/MindSpore/bin/python, python 3.7.10
- Ruta de instalación de paquetes de terceros: /home/ma-user/anaconda3/envs/MindSpore/lib/python3.7/site-packages
- Ciertos paquetes de instalación de pip:

```
cycler                          0.11.0
easydict                        1.9
Flask                           2.0.1
grpcio                          1.47.0
gunicorn                        20.1.0
ipykernel                       6.7.0
Jinja2                          3.0.1
lxml                            4.9.0
matplotlib                      3.5.1
mindinsight                     1.7.0
mindspore                       1.7.0
mindvision                      0.1.0
moxing-framework                2.1.0.5d9c87c8
numpy                           1.17.0
opencv-contrib-python-headless  4.6.0.66
opencv-python-headless          4.6.0.66
pandas                          1.1.5
Pillow                          9.1.1
pip                             22.1.2
protobuf                        3.20.1
psutil                          5.8.0
PyYAML                          5.1
requests                        2.27.1
scikit-learn                    0.22.1
scipy                           1.5.2
setuptools                      62.6.0
sklearn                         0.0
tensorboardX                    2.0
threadpoolctl                   3.1.0
tomli                           2.0.1
tornado                         6.1
tqdm                            4.64.0
traitlets                       5.3.0
treelib                         1.6.1
urllib3                         1.26.9
wheel                           0.37.1
zipp                            3.8.0
...
```

- Ciertos paquetes de instalación de apt:

```
apt
ca-certificates
cmake
cuda
```



```
curl
ethtool
fdisk
ffmpeg
g++
gcc
git
gpg
graphviz
libsm6
libxext6
libopencv-dev
libxrender-dev
libatlas3-base
libnuma-dev
libcap-dev
libssl-dev
liblz-dev
libbz2-dev
liblzma-dev
libboost-graph-dev
libsndfile1
libcurl4-openssl-dev
libopenblas-base
liblapack3
libopenblas-dev
libprotobuf-dev
libleveldb-dev
libsnappy-dev
libhdf5-serial-dev
liblapacke-dev
libgflags-dev
libgoogle-glog-dev
liblmdb-dev
libatlas-base-dev
librdmacm1
libcap2-bin
libpq-dev
mysql-common
net-tools
nginx
openslide-tools
openssh-client
openssh-server
openssh-sftp-server
openssl
protobuf-compiler
redis-server
redis-tools
rpm
tar
tofrodos
unzip
vim
wget
zip
zlib1g-dev
...
```


5 Preguntas frecuentes

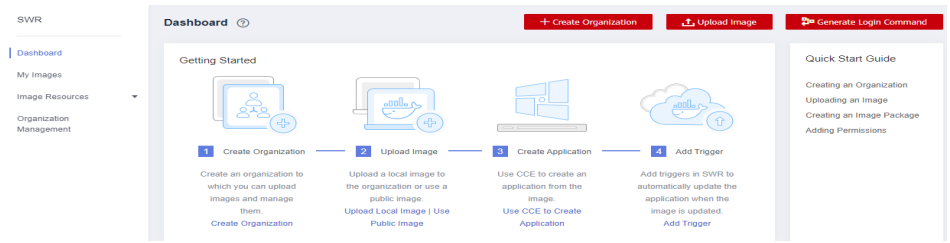
5.1 ¿Cómo puedo iniciar sesión en SWR y cargar imágenes en él?

Esta sección describe cómo iniciar sesión en SWR y cargar imágenes en él.

Paso 1 Iniciar sesión en SWR

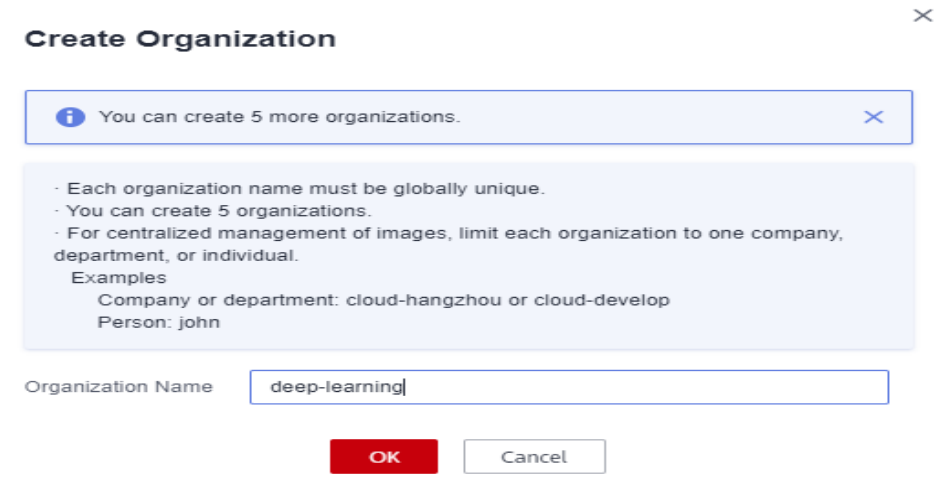
1. Inicie sesión en la consola de SWR y seleccione la región de destino.

   Figura 5-1 Consola de SWR

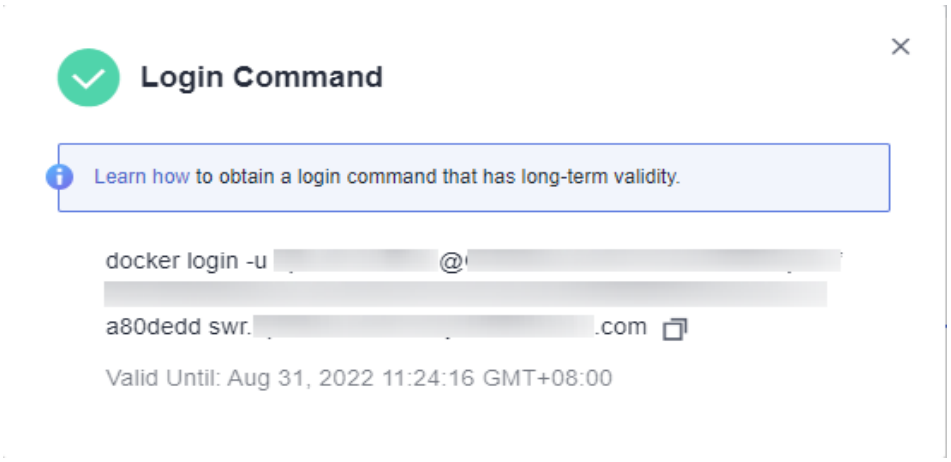
2. Haga clic en **Create Organization** en la esquina superior derecha e introduzca un nombre de organización para crear una organización. Se utiliza **deep-learning** como ejemplo. Sustitúyalo en los comandos siguientes por el nombre real de la organización.

Figura 5-2 Creación de una organización



3. Haga clic en **Generate Login Command** en la esquina superior derecha para obtener un comando de acceso.

Figura 5-3 Comando de acceso



4. Inicie sesión en el ECS como usuario **root** e ingrese el comando de acceso.

Figura 5-4 Comando de acceso ejecutado en el ECS

Paso 2 Subir imágenes a SWR

Esta sección describe cómo subir una imagen a SWR.

1. Inicie sesión en SWR y etiquete la imagen que se va a cargar. Reemplace el nombre de organización **deep-learning** en el siguiente comando con el nombre de organización real obtenido en el paso 1.
```
sudo docker tag tf-1.13.2:latest swr.xxx.com/deep-learning/tf-1.13.2:latest
```
2. Ejecute el siguiente comando para subir la imagen:
```
sudo docker push swr.xxx.com/deep-learning/tf-1.13.2:latest
```


Figura 5-5 Subir una imagen

3. Después de cargar la imagen, elija **My Images** en el panel de navegación a la izquierda de la consola de SWR para ver las imágenes personalizadas cargadas.

Figura 5-6 Imagen personalizada cargada

swr.xxx.com/deep-learning/tf-1.13.2:latest es el URL de SWR de la imagen personalizada.

5.2 ¿Cómo configuro variables de entorno para una imagen?

En un Dockerfile, utilice la instrucción ENV para configurar las variables de entorno. Para obtener más información, consulte la **referencia de Dockerfile**.

5.3 ¿Cómo inicio un contenedor usando una imagen de Docker?

Hay un punto de entrada en una imagen guardada, como se muestra en la figura siguiente. El punto de entrada no está preestablecido en la imagen. Por lo tanto, antes de ejecutar **docker run** para iniciar una imagen guardada, configure **entrypoint** para la entrada de inicio.

Para ello, ejecute el comando siguiente:

```
docker run -it -d --entrypoint=bash image:tag
```


Figura 5-7 Punto de entrada