ModelArts

Prácticas recomendadas de ModelArts

Edición 01

Fecha 2024-09-20

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Marcas registradas y permisos

El logotipo HUAWEI 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

Es posible que la totalidad o parte de los productos, las funcionalidades y/o los servicios que figuran en el presente documento no se encuentren dentro del alcance de un contrato vigente entre Huawei Cloud y el cliente. Las funcionalidades, los productos y los servicios adquiridos se limitan a los estipulados en el respectivo contrato. A menos que un contrato especifique lo contrario, ninguna de las afirmaciones, informaciones ni recomendaciones contenidas en el presente documento constituye garantía alguna, ni expresa ni implícita.

Huawei está permanentemente preocupada por la calidad de los contenidos de este documento; sin embargo, ninguna declaración, información ni recomendación aquí contenida constituye garantía alguna, ni expresa ni implícita. La información contenida en este documento se encuentra sujeta a cambios sin previo aviso.

Huawei Cloud Computing Technologies Co., Ltd.

Dirección: Huawei Cloud Data Center Jiaoxinggong Road
Avenida Qianzhong
Nuevo distrito de Gui'an
Gui Zhou, 550029
República Popular China

Sitio web: https://www.huaweicloud.com/intl/es-us/

Índice

1 Muestras oficiales

Este documento proporciona ejemplos de ModelArts sobre una variedad de escenarios y motores de IA para ayudarle a comprender rápidamente el proceso y las operaciones de usar ModelArts para el desarrollo de IA.

Permisos de ModelArts (Básico)

Muestra	Función	Escenario	Descripción
Escenarios	Permisos de IAM y configuraciones globales	Asignación de permisos para usuarios de IAM	Asignar los permisos de operación de ModelArts específicos a los usuarios de IAM bajo una cuenta de Huawei Cloud. Esto evita que se produzcan excepciones debidas a los permisos cuando los usuarios IAM acceden a ModelArts.

Muestras de algoritmos personalizados en desarrollo de modelos (avanzado)

Tabla 1-1 Muestras de algoritmos personalizados

Muestra	Imagen	Función	Escenario	Descripción
Uso de un algoritmo personalizado para crear un modelo de reconocimiento de dígitos escrito a mano	PyTorch	Personalización de algoritmos	Reconocimiento de dígitos manuscritos	Utilice su algoritmo personalizado para entrenar un modelo de reconocimiento de dígitos manuscritos y desplegar el modelo para la predicción.

2 Gestión de permisos

2.1 Conceptos básicos

ModelArts le permite configurar permisos de grano fino para una gestión refinada de recursos y permisos. Esto lo utilizan habitualmente las grandes empresas, pero es complejo para los usuarios individuales. Se recomienda que los usuarios individuales configuren los permisos para utilizar ModelArts según **Asignación de permisos a los usuarios individuales para usar ModelArts**.

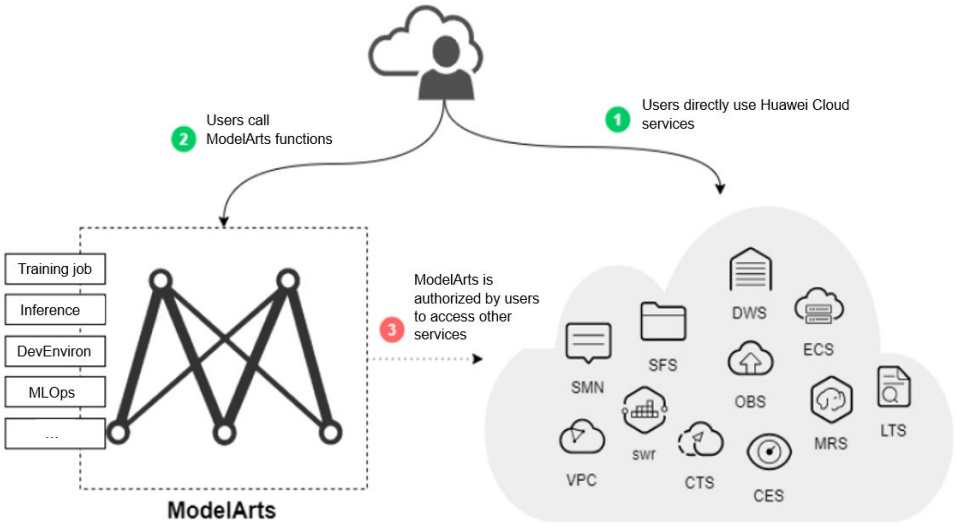
> **NOTA**
>
> Si cumple alguna de las siguientes condiciones, lea este documento.
>
> - Usted es un usuario empresarial y
> - En su empresa hay varios departamentos y necesita controlar los permisos de los usuarios para que los usuarios de los distintos departamentos sólo puedan acceder a los recursos y funciones que les corresponden.
> - Existen múltiples roles en su empresa, incluidos administradores, desarrolladores de algoritmos e ingenieros de O&M de aplicaciones. Solo es necesario que utilicen las funciones específicas.
> - Lógicamente, existen varios entornos aislados, como el entorno de desarrollo, el entorno de preproducción y el entorno de producción. Es necesario controlar los permisos de los usuarios en diferentes entornos.
> - Es necesario controlar los permisos de usuarios o grupos de usuarios específicos de IAM.
> - Usted es un usuario individual y ha creado varios usuarios de IAM. Debe asignar los permisos diferentes de ModelArts a diferentes usuarios de IAM.
> - Debe comprender los conceptos y operaciones de la gestión de permisos de ModelArts.

ModelArts utiliza Identity and Access Management (IAM) para la mayoría de los permisos de las funciones de gestión. Antes de leer a continuación, conozca los **Conceptos básicos**. Esto le ayudará a comprender mejor este documento.

Para implementar una gestión de permisos detallada, ModelArts proporciona control de permisos, autorización de agencias y espacio de trabajo. A continuación se describen los detalles.

Permisos y delegaciones de ModelArts

Figura 2-1 Gestión de permisos



Las funciones expuestas de ModelArts se controlan mediante permisos de IAM. Por ejemplo, si usted, como usuario de IAM, necesita crear un trabajo de entrenamiento en ModelArts, debe tener el permiso **ModelArts:trainJob:create**. Para obtener detalles sobre cómo asignar permisos a un usuario (necesita agregar el usuario a un grupo de usuarios y luego asignar los permisos a este grupo), consulte **Gestión de permisos**.

ModelArts debe acceder a otros servicios para el cómputo de IA. Por ejemplo, ModelArts debe acceder al OBS para leer los datos para el entrenamiento. Por razones de seguridad, ModelArts debe estar autorizado para acceder a otros servicios en la nube. Esta es la autorización de la delegación.

A continuación se presenta un resumen de la gestión de permisos:

- El acceso a cualquier servicio en la nube se controla con IAM. Debe tener los permisos del servicio en la nube. (Los permisos de servicios requeridos varían según las funciones que utilice)
- Para utilizar las funciones de ModelArts, debe otorgar permisos con IAM.
- ModelArts debe estar autorizado por usted para acceder a otros servicios en la nube para el cómputo de IA.

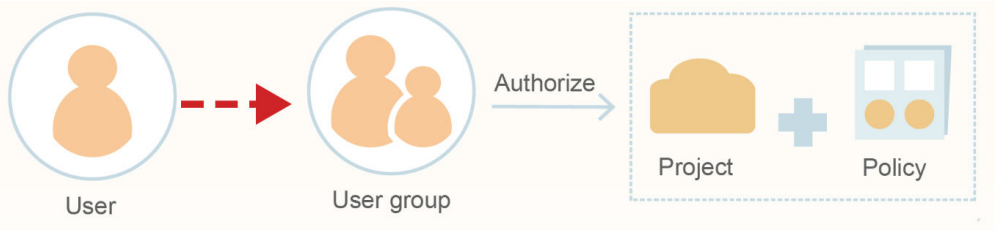
Gestión de permisos de ModelArts

De forma predeterminada, los nuevos usuarios de IAM no tienen ningún permiso asignado. Es necesario agregar el usuario a un grupo de usuarios y otorgar políticas al grupo de usuarios, de modo que los usuarios del grupo puedan heredar los permisos. Después de la autorización, los usuarios pueden realizar operaciones de ModelArts basadas en permisos.

⚠ ATENCIÓN

ModelArts es un servicio a nivel de proyecto desplegado y al que se accede en regiones físicas específicas. Cuando autoriza una delegación, puede establecer el alcance de los permisos seleccionados para todos los recursos, proyectos empresariales o proyectos específicos de la región. Si especifica proyectos específicos para cada región, los permisos seleccionados se aplicarán a los recursos de estos proyectos.

Para obtener más detalles, consulte **Creación de un grupo de usuarios y asignación de permisos**.



Al asignar permisos a un grupo de usuarios, IAM no asigna directamente los permisos específicos al grupo de usuarios. En su lugar, IAM debe agregar los permisos a una política y luego asignar la política al grupo de usuarios. Para facilitar la gestión de los permisos de usuarios, cada servicio en la nube ofrece algunas políticas preestablecidas para que usted las utilice directamente. Si las políticas preestablecidas no pueden satisfacer sus requisitos de gestión de permisos de grano fino, puede personalizar las políticas.

Tabla 2-1 enumera todas las políticas preestablecidas definidas por el sistema que ModelArts admite.

Tabla 2-1 Políticas definidas por el sistema compatibles con ModelArts

Política	Descripción	Tipo
ModelArts FullAccess	Permisos de administrador para ModelArts. Los usuarios a los que se les conceden estos permisos pueden operar y usar ModelArts.	Política definida por el sistema
ModelArts CommonOperations	Permisos de usuario comunes para ModelArts. Los usuarios a los que se conceden estos permisos pueden operar y usar ModelArts pero no pueden gestionar grupos de recursos dedicados.	Política definida por el sistema
ModelArts Dependency Access	Permisos en los servicios dependientes para ModelArts	Política definida por el sistema

Por lo general, ModelArts FullAccess solo se asigna a los administradores. Si no se requiere una gestión detallada, asignar ModelArts CommonOperations a todos los usuarios cumplirá con los requisitos de desarrollo de la mayoría de los equipos pequeños. Si desea personalizar políticas para una gestión detallada, consulte **IAM**.

> **NOTA**
>
> Cuando se asigna permisos de ModelArts a un usuario, el sistema no asigna automáticamente los permisos de otros servicios al usuario. Esto garantiza la seguridad y evita las operaciones no autorizadas inesperadas. En este caso, sin embargo, debe asignar por separado permisos de diferentes servicios a los usuarios para que puedan realizar algunas operaciones de ModelArts.
>
> Por ejemplo, si un usuario de IAM necesita utilizar datos de OBS para entrenamiento y el permiso de entrenamiento de ModelArts ha sido configurado para el usuario de IAM, el usuario de IAM aún necesita ser asignado con los permisos de OBS de lectura, escritura y lista. El permiso de lista de OBS permite seleccionar la ruta de datos de entrenamiento en ModelArts. El permiso de lectura se utiliza para obtener una vista previa de los datos y leer datos para el entrenamiento. El permiso de escritura se utiliza para guardar los resultados y logs del entrenamiento.
>
> - Para usuarios individuales o pequeñas organizaciones, es una buena práctica configurar la política **Tenant Administrator** que se aplica a los servicios globales para usuarios de IAM. De esta manera, los usuarios de IAM pueden obtener todos los permisos de usuario excepto IAM. Sin embargo, esto puede causar problemas de seguridad. (Para un usuario individual, su usuario de IAM predeterminado pertenece al grupo de usuarios **admin** y tiene el permiso **Tenant Administrator**.)
> - Si desea restringir las operaciones de usuarios, configure los permisos mínimos de OBS para los usuarios de ModelArts. Para obtener más detalles, consulte **Gestión de permisos de OBS**. Para obtener más información sobre la gestión de permisos detallados de otros servicios en la nube, consulte los documentos de los servicios en la nube correspondientes.

Autorización de delegación de ModelArts

ModelArts debe estar autorizado por los usuarios para acceder a otros servicios en la nube para el cómputo de IA. En el sistema de permisos de IAM, dicha autorización se realiza con delegaciones.

Para obtener más información sobre los conceptos básicos y las operaciones de delegaciones, consulte **Delegación de servicios en la nube**.

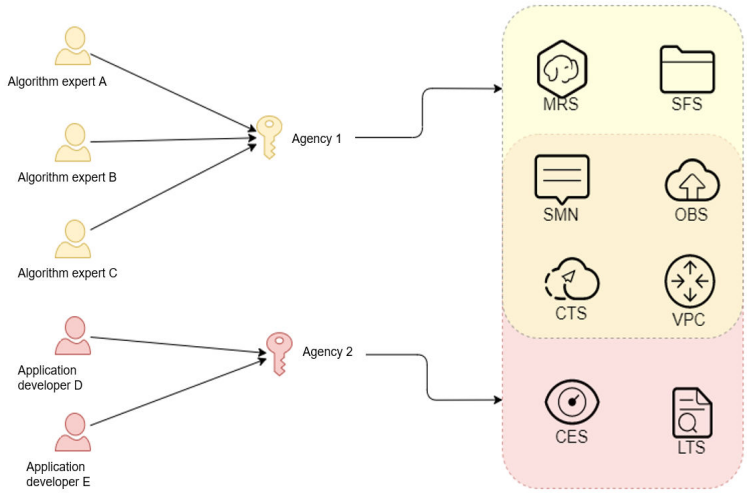
Para simplificar la autorización de delegaciones, ModelArts admite la configuración automática de autorizaciones de delegaciones. Solo necesita configurar una delegación para usted o para los usuarios especificados en la página **Global Configuration** de la consola de ModelArts.

> **NOTA**
>
> - Solo los usuarios con permiso de gestión de agencias de IAM pueden realizar esta operación. Por lo general, los miembros del grupo de usuarios admin de IAM tienen este permiso.
> - La autorización de delegación de ModelArts es específica de cada región, lo que significa que debe realizar la autorización de delegación en cada región que utilice.

En la página **Global Configuration** de la consola de ModelArts, después de hacer clic en **Add Authorization**, puede configurar una delegación para un usuario específico o para todos los usuarios. Por lo general, se crea una delegación denominada **modelarts_agency_<***Username***>_***Random ID*** de forma predeterminada. En el área **Permissions**, puede seleccionar la configuración de permisos preestablecida o seleccionar las políticas necesarias. Si ambas opciones no satisfacen sus requisitos, puede crear una delegación en la página de gestión de IAM (necesita delegar ModelArts para acceder a sus recursos) y luego usar una delegación existente en lugar de agregar una delegación en la página **Add Authorization**.

ModelArts asocia varios usuarios con una delegación. Esto significa que si dos usuarios necesitan configurar la misma delegación, no es necesario crear una delegación para cada usuario. En su lugar, solo necesita configurar la misma delegación para los dos usuarios.

Figura 2-2 Mapeo entre usuarios y agencias



NOTA

Cada usuario puede utilizar ModelArts solo después de asociarse con una delegación. Sin embargo, aunque los permisos asignados a la delegación sean insuficientes, no se reporta ningún error cuando se invoca a la API. Solo se produce un error cuando el sistema utiliza funciones no autorizadas. Por ejemplo, puede activar la notificación de mensajes al crear un trabajo de entrenamiento. La notificación de mensajes requiere autorización de SMN. Sin embargo, solo se produce un error cuando es necesario enviar mensajes para el trabajo de entrenamiento. El sistema ignora algunos errores y otros pueden causar fallas en el trabajo. Cuando implemente la minimización de permisos, asegúrese de que aún tendrá suficientes permisos para las operaciones requeridas en ModelArts.

Autorización estricta

En el modo de autorización estricta, se requiere la autorización explícita por el administrador de la cuenta para que los usuarios de IAM puedan acceder a ModelArts. El administrador puede agregar los permisos de ModelArts requeridos a los usuarios comunes mediante las políticas de autorización.

En el modo de autorización no estricta, los usuarios de IAM pueden usar ModelArts sin autorización explícita. El administrador necesita configurar la política de denegación para usuarios de IAM para evitar que utilicen algunas funciones de ModelArts.

El administrador puede cambiar el modo de autorización en la página **Global Configuration**.

AVISO

Se recomienda el modo de autorización estricta. En este modo, los usuarios de IAM deben estar autorizados a utilizar funciones de ModelArts. De este modo, se puede controlar con precisión el alcance de los permisos de los usuarios de IAM, minimizando los permisos concedidos a los usuarios de IAM.

Gestión del acceso a los recursos con espacios de trabajo

El espacio de trabajo permite a los clientes empresariales dividir sus recursos en varios espacios lógicamente aislados y gestionar el acceso a los distintos espacios. Como usuario de

empresa, puede enviar la solicitud de habilitación de la función de espacio de trabajo a su responsable de asistencia técnica.

Una vez que se habilita el espacio de trabajo, se crea un espacio de trabajo predeterminado. Todos los recursos que ha creado se encuentran en este espacio de trabajo. Un espacio de trabajo es como un gemelo de ModelArts. Puede cambiar de un espacio de trabajo a otro en la esquina superior izquierda de la consola de ModelArts. Los trabajos en espacios de trabajo diferentes no se afectan entre sí.

Al crear un espacio de trabajo, debe vincularlo a un proyecto empresarial. Pueden vincularse múltiples espacios de trabajo a un mismo proyecto empresarial, pero no puede vincularse un espacio de trabajo a varios proyectos empresariales. Puede utilizar espacios de trabajo para refinar las restricciones de acceso a los recursos y los permisos de los distintos usuarios. Las restricciones son las siguientes:

- Los usuarios deben estar autorizados para acceder a espacios de trabajo específicos (esto debe configurarse en las páginas de creación y gestión de espacios de trabajo). Esto significa que el acceso a activos de IA, como conjuntos de datos y algoritmos, puede gestionarse mediante espacios de trabajo.
- En las operaciones de autorización de permisos anteriores, si establece el ámbito en proyectos de empresa, la autorización solo tendrá efecto para los espacios de trabajo vinculados a los proyectos seleccionados.

NOTA

- Las restricciones sobre los espacios de trabajo y la autorización de permisos surten efecto al mismo tiempo. Es decir, un usuario debe tener los permisos para acceder al espacio de trabajo y para crear trabajos de entrenamiento (el permiso se aplica a este espacio de trabajo) para que el usuario pueda enviar trabajos de entrenamiento en este espacio de trabajo.
- Si ha habilitado un proyecto de empresa pero no ha habilitado un espacio de trabajo, todas las operaciones se realizarán en el proyecto de empresa predeterminado. Asegúrese de que los permisos de las operaciones requeridas se aplican al proyecto de empresa por defecto.
- Las restricciones anteriores no se aplican a los usuarios que no hayan habilitado ningún proyecto de empresa.

Resumen

Características principales de la gestión de permisos de ModelArts:

- Si usted es un usuario individual, no es necesario que considere la gestión detallada de permisos. Su cuenta tiene todos los permisos para utilizar ModelArts de forma predeterminada.
- IAM controla todas las funciones de ModelArts. Puede utilizar la autorización de IAM para implementar una gestión detallada de permisos para los usuarios específicos.
- Todos los usuarios (incluidos los usuarios individuales) pueden usar funciones específicas solo después de la autorización de la delegación en ModelArts (**Settings** > **Add Authorization**). De lo contrario, pueden producirse errores inesperados.
- Si ha habilitado la función de proyecto empresarial, también puede habilitar el espacio de trabajo de ModelArts y utilizar tanto la autorización básica como el espacio de trabajo para una gestión de permisos refinada.

2.2 Mecanismos de gestión de permiso

2.2.1 IAM

Esta sección describe las configuraciones de permisos de IAM para todas las funciones de ModelArts.

Permisos de IAM

Si necesita asignar diferentes permisos a los empleados de su empresa para acceder a sus recursos de ModelArts, Identity and Access Management (IAM) es una buena opción para la gestión de permisos detallada. IAM proporciona autenticación de identidad, gestión de permisos y control de acceso, lo que le ayuda a acceder de forma segura a los recursos de Huawei Cloud. Si su cuenta de Huawei puede cumplir con sus requerimientos y no necesita una cuenta de IAM para gestionar los permisos de usuarios, omita este capítulo.

IAM es un servicio gratuito. Solo paga por los recursos de su cuenta.

Con IAM, puede controlar el acceso a recursos específicos de Huawei Cloud. Por ejemplo, si los desarrolladores de software en su empresa necesitan poseer permisos para usar ModelArts, pero usted no quiere que posean permisos de operación de alto riesgo como borrar ModelArts, usted puede otorgar permisos usando IAM para limitar sus permisos en ModelArts.

Para obtener más información acerca de IAM, consulte **Qué es IAM**.

Autorización basada en roles/políticas

ModelArts admite la autorización basada en roles/políticas. De forma predeterminada, los nuevos usuarios de IAM no tienen permisos. Debe agregar un usuario a uno o más grupos y asignar políticas de permisos o roles a estos grupos. Los usuarios heredan los permisos de los grupos a los que se agregan. Este proceso se llama autorización. A continuación, los usuarios heredan los permisos de los grupos y pueden realizar las operaciones especificadas en los servicios en nube.

ModelArts es un servicio a nivel de proyecto desplegado para las regiones específicas. Al establecer **Scope** en **Region-specific projects** y seleccionar los proyectos especificados (por ejemplo, **ap-southeast-2**) en las regiones especificadas (por ejemplo, **AP-Bangkok**), los usuarios solo tienen permisos para los recursos de ModelArts en los proyectos seleccionados. Si establece **Scope** en **All resources**, los usuarios dispondrán de recursos de ModelArts en todos los proyectos específicos de la región. Al acceder a ModelArts los usuarios deben cambiar a una región en la que han sido autorizados a usar servicios en la nube.

Tabla 2-2 enumera todas las políticas definidas por el sistema admitidas por ModelArts. Si el permiso de ModelArts preestablecido no cumple con sus requerimientos, cree una política personalizada consultando **Campos de políticas en formato JSON**.

Tabla 2-2 Políticas definidas por el sistema compatibles con ModelArts

Política	Descripción	Tipo
ModelArts FullAccess	Todos los permisos para administradores de ModelArts	Política definida por el sistema
ModelArts CommonOperations	Todos los permisos de operación para los usuarios comunes de ModelArts, lo que no incluye la gestión de grupos de recursos dedicados.	Política definida por el sistema

Política	Descripción	Tipo
ModelArts Dependency Access	Permisos en los servicios dependientes para ModelArts	Política definida por el sistema

ModelArts depende de otros servicios en la nube. Para verificar o consultar los servicios en la nube, configure los permisos correspondientes en la consola de ModelArts, como se muestra en la siguiente tabla.

Tabla 2-3 Dependencias de roles/políticas de la consola de ModelArts

Función de consola	Dependencia	Rol/política requerido
Gestión de datos	Object Storage Service (OBS)	OBS Administrator
	Data Lake Insight (DLI)	DLI FullAccess
	MapReduce Service (MRS)	MRS Administrator
	GaussDB(DWS)	DWS Administrator
	Cloud Trace Service (CTS)	CTS Administrator
	ModelArts	ModelArts CommonOperations ModelArts Dependency Access
Entorno de desarrollo	OBS	OBS Administrator
	Cloud Secret Management Service (CSMS)	CSMS ReadOnlyAccess
	CTS	CTS Administrator
	Elastic Cloud Server (ECS)	ECS FullAccess
	Software Repository for Container (SWR)	SWR Administrator
	Scalable File Service (SFS)	SFS Turbo FullAccess
	Application Operations Management (AOM)	AOM FullAccess
	Key Management Service (KMS)	KMS CMKFullAccess

Función de consola	Dependencia	Rol/política requerido
	ModelArts	ModelArts CommonOperations ModelArts Dependency Access
Gestión de entrenamientos	OBS	OBS Administrator
	Simple Message Notification (SMN)	SMN Administrator
	CTS	CTS Administrator
	SFS Turbo	SFS Turbo ReadOnlyAccess
	SWR	SWR Administrator
	AOM	AOM FullAccess
	KMS	KMS CMKFullAccess
	ModelArts	ModelArts CommonOperations ModelArts Dependency Access
Flujo de trabajo	OBS	OBS Administrator
	CTS	CTS Administrator
	ModelArts	ModelArts CommonOperations ModelArts Dependency Access
ExeML	OBS	OBS Administrator
	CTS	CTS Administrator
	ModelArts	ModelArts CommonOperations ModelArts Dependency Access
Gestión de aplicaciones de IA	OBS	OBS Administrator
	Enterprise Project Management Service (EPS)	EPS FullAccess
	CTS	CTS Administrator
	SWR	SWR Administrator
	ModelArts	ModelArts CommonOperations ModelArts Dependency Access
Despliegue del servicio	OBS	OBS Administrator
	Cloud Eye Service (CES)	CES ReadOnlyAccess
	SMN	SMN Administrator

Función de consola	Dependencia	Rol/política requerido
	EPS	EPS FullAccess
	CTS	CTS Administrator
	Log Tank Service (LTS)	LTS FullAccess
	Virtual Private Cloud (VPC)	VPC FullAccess
	ModelArts	ModelArts CommonOperations ModelArts Dependency Access
AI Gallery	OBS	OBS Administrator
	CTS	CTS Administrator
	SWR	SWR Administrator
	ModelArts	ModelArts CommonOperations ModelArts Dependency Access
Grupo de recursos dedicados	CTS	CTS Administrator
	Cloud Container Engine (CCE)	CCE Administrator
	Bare Metal Server (BMS)	BMS FullAccess
	Image Management Service (IMS)	IMS FullAccess
	Data Encryption Workshop (DEW)	DEW KeypairReadOnlyAccess
	VPC	VPC FullAccess
	ECS	ECS FullAccess
	SFS	SFS Turbo FullAccess
	OBS	OBS Administrator
	AOM	AOM FullAccess
	ModelArts	ModelArts FullAccess
	Billing Center	BSS Administrator

Si las políticas definidas por el sistema no pueden satisfacer sus requisitos, puede crear una política personalizada. Para obtener más información sobre las acciones admitidas por las directivas personalizadas, consulte **Permisos de recursos de ModelArts**.

Puede crear las políticas personalizadas de cualquiera de las siguientes maneras:

- Visual editor: Seleccione los servicios en la nube, acciones, recursos y condiciones de solicitud sin la necesidad de conocer la sintaxis de la política.
- JSON: Crea una política JSON o edita una existente.

Para obtener más información, consulte **Creación de una política personalizada**. A continuación se enumeran ejemplos de políticas personalizadas comunes de ModelArts.

- Ejemplo 1: Conceder permiso para gestionar imágenes.

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "modelarts:image:register",
                "modelarts:image:listGroup"
            ]
        }
    ]
}
```

- Ejemplo 2: Conceder permiso para denegar la creación, actualización y eliminación de un grupo de recursos dedicados.

  Una política con solo permisos "Deny" debe usarse junto con otras políticas. Si los permisos concedidos a un usuario de IAM contienen tanto "Allow" como "Deny", los permisos "Deny" tienen la prioridad sobre los permisos "Allow".

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Action": [
                "modelarts:*:*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "swr:*:*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "smn:*:*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "modelarts:pool:create",
                "modelarts:pool:update",
                "modelarts:pool:delete"
            ],
            "Effect": "Deny"
        }
    ]
}
```

- Ejemplo 3: Crear una política personalizada que contenga varias acciones.

  Una política personalizada puede contener acciones de varios servicios que son de tipo global o de nivel de proyecto. A continuación se muestra una política de ejemplo que contiene acciones de varios servicios:

```
{
    "Version": "1.1",
    "Statement": [
```



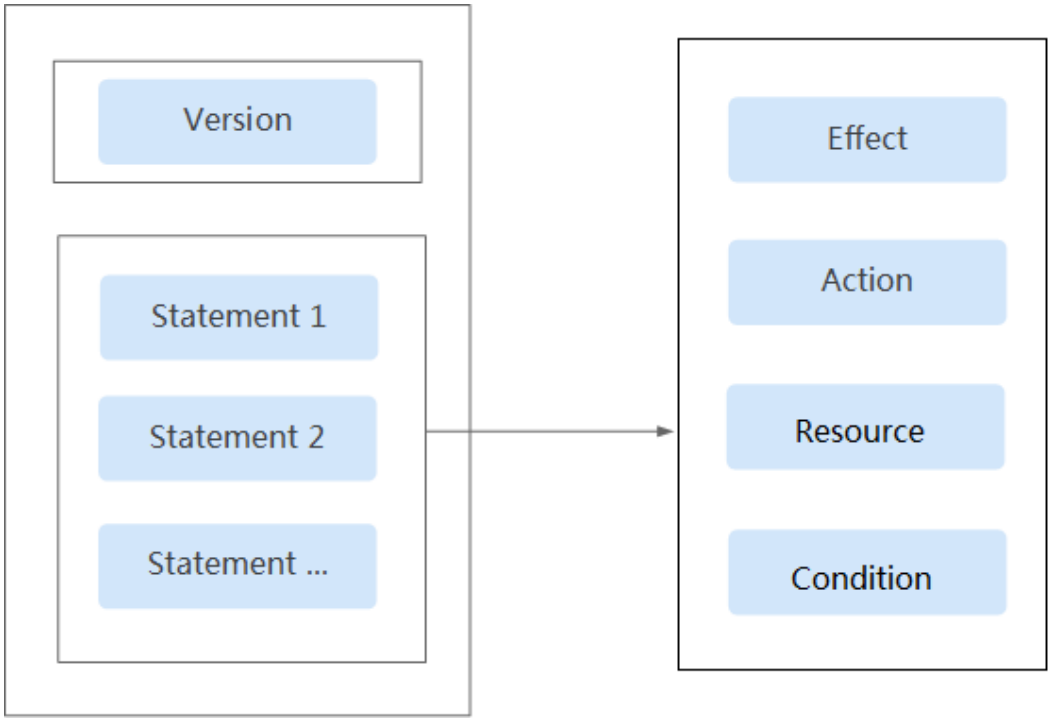
```
        {
            "Effect": "Allow",
            "Action": [
                "modelarts:service:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lts:logs:list"
            ]
        }
    ]
}
```

Campos de políticas en formato JSON

Estructura de políticas

Una política consiste en una versión y una o más sentencias (que indican diferentes acciones).

Figura 2-3 Estructura de políticas



Parámetros de políticas

A continuación se describen los parámetros de política. Puede crear políticas personalizadas especificando los parámetros. Para obtener más información, consulte **Casos de uso de políticas personalizadas**.

Tabla 2-4 Parámetros de política

Parámetro		Descripción	Valor
Version		Versión de política	**1.1**: indica el control de acceso basado en políticas.
Statement: authorization statement of a policy	Effect	Permitir o denegar las operaciones definidas en la acción	• **Allow**: indica que la operación está permitida. • **Deny**: indica que la operación no está permitida. **NOTA** Si la política utilizada para otorgar los permisos de usuario contiene **Allow** y **Deny** para la misma acción, **Deny** tiene prioridad.
	Action	Operación a realizar en el servicio	Formato: *nombre de servicio*:*Tipo de recurso*:*Acción*. Se admiten caracteres carácter comodín (*), que indican todas las opciones. Ejemplo: **modelarts:notebook:list**: indica el permiso para ver una lista de instancias de notebook. **modelarts** indica el nombre del servicio, **notebook** indica el tipo de recurso y **list**, la operación. Ver todas las acciones de un servicio en su *Referencia de las API*.
	Condition	Condición para que una política entre en vigor, incluidas **las claves de condición y los operadores**	Formato: "*Condition operator*:{*Condition key*:[*Value 1*,*Value 2*]}" Si establece varias condiciones, la política solo tendrá efecto cuando se cumplan todas las condiciones. Ejemplo: **StringEndWithIfExists":{"g:UserName":["specialCharacter"]}**: La instrucción es válida para los usuarios cuyos nombres terminan con **specialCharacter**.
	Resource	Recursos sobre los que entra en vigor una política	Formato: *Service name*:<*Region*>:<*Account ID*>:*Resource type*:*Resource path*. Se soportan asteriscos (*) para el tipo de recurso, que indican todos los recursos. **NOTA** La autorización de ModelArts no permite especificar una ruta de acceso de recurso.

Tipos de recursos de ModelArts

Los administradores pueden especificar el ámbito en función de los tipos de recursos de ModelArts. En la siguiente tabla se enumeran los tipos de recursos admitidos por ModelArts:

Tabla 2-5 Tipos de recursos soportados por la autorización según rol/políticas de ModelArts

Tipo de recurso	Descripción
notebook	Instancias de notebook de DevEnviron
exemlProject	Proyectos de ExeML
exemlProjectInf	Servicio de inferencia en tiempo real desarrollado por ExeML
exemlProjectTrain	Trabajos de entrenamiento desarrollado por ExeML
exemlProjectVersion	Versión del proyecto de ExeML
workflow	Flujo de trabajo
pool	Grupo de recursos dedicados
network	Conexión en red de un grupo de recursos dedicado
trainJob	Trabajo de entrenamiento
trainJobLog	Registros de tiempo de ejecución de un trabajo de entrenamiento
trainJobInnerModel	Modelo predeterminado
trainJobVersion	Versión de un trabajo de entrenamiento (con el apoyo de trabajos de entrenamiento de versiones antiguas que se suspenderán en breve)
trainConfig	Configuración de un trabajo de entrenamiento (con el apoyo de trabajos de entrenamiento de versiones antiguas que se suspenderán en breve)
tensorboard	Trabajo de visualización de los resultados del entrenamiento (con el apoyo de trabajos de entrenamiento de versiones antiguas que se suspenderán en breve)
model	Modelos
service	Servicio en tiempo real
nodeservice	Servicio en el borde
workspace	Espacio de trabajo
dataset	Conjunto de datos
dataAnnotation	Etiquetas de conjunto de datos
aiAlgorithm	Algoritmo para trabajos de entrenamiento
image	Imagen

Tipo de recurso	Descripción
devserver	El BMS elástico

Permisos de recursos de ModelArts

Para obtener más detalles, consulte «Políticas de permisos y acciones admitidas» en la *Referencia de las API de ModelArts*.

- **Permisos de gestión de datos**
- **Permisos de DevEnviron**
- **Permisos de trabajo de entrenamiento**
- **Permisos de gestión de modelos**
- **Permisos de gestión de servicios**

2.2.2 Delegaciones y dependencias

Dependencia de funciones

Políticas de la dependencia de funciones

Cuando utilice ModelArts para desarrollar algoritmos o gestionar trabajos de entrenamiento, deberá utilizar otros servicios en la nube. Por ejemplo, antes de enviar un trabajo de entrenamiento, seleccione una ruta de OBS para almacenar el conjunto de datos y los logs, respectivamente. Por lo tanto, al configurar políticas de autorización de grano fino para un usuario, el administrador debe configurar permisos dependientes para que el usuario pueda utilizar las funciones requeridas.

NOTA

Si utiliza ModelArts como usuario de root (el usuario de IAM por defecto con el mismo nombre que la cuenta), el usuario de root tiene todos los permisos predeterminados.

Tabla 2-6 Configuración básica

Escenario de aplicación	Servicio dependiente	Política dependiente	Función admitida
Configuración global	IAM	iam:users:listUsers	Obtener una lista de usuarios. Solo el administrador requiere esta acción.
Función básica	IAM	iam:tokens:assume	(Obligatorio) Utilice una delegación para obtener credenciales de autenticación temporales.
Función básica	BSS	bss:balance:view	Mostrar el saldo de la cuenta actual en la página después de crear los recursos en la consola ModelArts.

Tabla 2-7 Gestión de los espacios de trabajo

Escenario de aplicación	Servicio dependiente	Política dependiente	Función admitida
Espacio de trabajo	IAM	iam:users:listUsers	Autorizar a un usuario de IAM a utilizar un espacio de trabajo.
	ModelArts	modelarts:*:*delete*	Borrar los recursos en un espacio de trabajo cuando eliminarlo.

Tabla 2-8 Gestión de instancias de notebook

Escenario de aplicación	Servicio dependiente	Política dependiente	Función admitida
Gestión del ciclo de vida de las instancias del entorno de desarrollo	ModelArts	modelarts:notebook:create modelarts:notebook:list modelarts:notebook:get modelarts:notebook:update modelarts:notebook:delete modelarts:notebook:start modelarts:notebook:stop modelarts:notebook:updateStopPolicy modelarts:image:delete modelarts:image:list modelarts:image:create modelarts:image:get modelarts:pool:list modelarts:tag:list modelarts:network:get aom:metric:get aom:metric:list aom:alarm:list	Iniciar, detener, crear, eliminar y actualizar una instancia.

Escenario de aplicación	Servicio dependiente	Política dependiente	Función admitida
Almacenamiento de montaje dinámico	ModelArts	modelarts:notebook:listMountedStorages modelarts:notebook:mountStorage modelarts:notebook:getMountedStorage modelarts:notebook:umountStorage	Montar almacenamiento de forma dinámica.
	OBS	obs:bucket:ListAllMyBuckets obs:bucket:ListBucket	
Gestión de imágenes	ModelArts	modelarts:image:register modelarts:image:listGroup	Registre y consulte una imagen en la página **Image Management**.
Guardar una imagen	SWR	SWR Admin	La política **SWR Admin** contiene el alcance máximo de los permisos de SWR, que puede utilizarse para: ● Guarde una instancia del entorno de desarrollo en ejecución como imagen. ● Cree una instancia de notebook con una imagen personalizada.
Uso de la función de SSH	ECS	ecs:serverKeypairs:list ecs:serverKeypairs:get ecs:serverKeypairs:delete ecs:serverKeypairs:create	Configure una clave de inicio de sesión para una instancia de notebook.
Montaje de un sistema de archivos de SFS Turbo	SFS Turbo	SFS Turbo FullAccess	Leer y escribir un directorio de SFS como usuario de IAM. Monte un sistema de archivos de SFS que no haya creado usted en una instancia de notebook mediante un grupo de recursos dedicado.

Escenario de aplicación	Servicio dependiente	Política dependiente	Función admitida
Consulta de todas las instancias	ModelArts	modelarts:notebook:listAllNotebooks	Consultar instancias del entorno de desarrollo de todos los usuarios en la consola de gestión de ModelArts. Esta acción es requerida por el administrador de la instancia del entorno de desarrollo.
	IAM	iam:users:listUsers	
Complemento de VS Code local o PyCharm Toolkit	ModelArts	modelarts:notebook:listAllNotebooks modelarts:trainJob:create modelarts:trainJob:list modelarts:trainJob:update modelarts:trainJobVersion:delete modelarts:trainJob:get modelarts:trainJob:logExport modelarts:workspace:getQuotas (This policy is required if the **workspace** function is enabled.)	Acceda a una instancia de notebook desde VS Code local y envíe trabajos de entrenamiento.

Escenario de aplicación	Servicio dependiente	Política dependiente	Función admitida
	OBS	obs:bucket:ListAllMybuckets obs:bucket:HeadBucket obs:bucket:ListBucket obs:bucket:GetBucketLocation obs:object:GetObject obs:object:GetObjectVersion obs:object:PutObject obs:object:DeleteObject obs:object:DeleteObjectVersion obs:object:ListMultipartUploadParts obs:object:AbortMultipartUpload obs:object:GetObjectAcl obs:object:GetObjectVersionAcl obs:bucket:PutBucketAcl obs:object:PutObjectAcl obs:object:ModifyObjectMetaData	
	IAM	iam:projects:listProjects	Obtenga una lista de proyectos de IAM con PyCharm local para configuraciones de acceso.

Tabla 2-9 Gestión de trabajos de entrenamiento

Escenario de aplicación	Servicio dependiente	Política dependiente	Función admitida
Gestión de entrenamientos	ModelArts	modelarts:trainJob:* modelarts:trainJobLog:* modelarts:aiAlgorithm:* modelarts:image:list	Crear un trabajo de entrenamiento y consultar los logs de entrenamiento.

Escenario de aplicación	Servicio dependiente	Política dependiente	Función admitida
		modelarts:workspace:getQuotas	Obtener una cuota de espacio de trabajo. Esta política es necesaria si se habilita la función de **espacio de trabajo**.
		modelarts:tag:list	Utilice Tag Management Service (TMS) en un trabajo de entrenamiento.
	IAM	iam:credentials:listCredentials iam:agencies:listAgencies	Utilice la autorización de delegación configurada.
	SFS Turbo	sfsturbo:shares:getShare sfsturbo:shares:getAllShares	Utilizar SFS Turbo en un trabajo de entrenamiento.
	SWR	swr:repository:listTags swr:repository:getRepository swr:repository:listRepositories	Utilice una imagen personalizada para crear un trabajo de entrenamiento.
	SMN	smn:topic:publish smn:topic:list	Notificar cambios en el estado del trabajo de entrenamiento con SMN.

Escenario de aplicación	Servicio dependiente	Política dependiente	Función admitida
	OBS	obs:bucket:ListAllMybuckets obs:bucket:HeadBucket obs:bucket:ListBucket obs:bucket:GetBucketLocation obs:object:GetObject obs:object:GetObjectVersion obs:object:PutObject obs:object:DeleteObject obs:object:DeleteObjectVersion obs:object:ListMultipartUpload-Parts obs:object:AbortMultipartUpload obs:object:GetObjectAcl obs:object:GetObjectVersionAcl obs:bucket:PutBucketAcl obs:object:PutObjectAcl obs:object:ModifyObjectMetaData	Ejecutar un trabajo de entrenamiento con un conjunto de datos en un bucket de OBS.

Tabla 2-10 Uso de flujos de trabajo

Escenario de aplicación	Servicio dependiente	Política dependiente	Función admitida
Uso de un conjunto de datos	ModelArts	modelarts:dataset:getDataset modelarts:dataset:createDataset modelarts:dataset:createDatasetVersion modelarts:dataset:createImportTask modelarts:dataset:updateDataset modelarts:processTask:createProcessTask modelarts:processTask:getProcessTask modelarts:dataset:listDatasets	Utilice conjuntos de datos de ModelArts en un flujo de trabajo.

Escenario de aplicación	Servicio dependiente	Política dependiente	Función admitida
Gestión de aplicaciones de IA	ModelArts	modelarts:model:list modelarts:model:get modelarts:model:create modelarts:model:delete modelarts:model:update	Gestione aplicaciones de IA de ModelArts en un flujo de trabajo.
Despliegue de un servicio	ModelArts	modelarts:service:get modelarts:service:create modelarts:service:update modelarts:service:delete modelarts:service:getLogs	Gestione servicios de ModelArts en tiempo real en un flujo de trabajo.
Trabajos de entrenamiento	ModelArts	modelarts:trainJob:get modelarts:trainJob:create modelarts:trainJob:list modelarts:trainJobVersion:list modelarts:trainJobVersion:create modelarts:trainJob:delete modelarts:trainJobVersion:delete modelarts:trainJobVersion:stop	Gestione los trabajos de entrenamiento de ModelArts en un flujo de trabajo.
Espacio de trabajo	ModelArts	modelarts:workspace:get modelarts:workspace:getQuotas	Utilizar los espacios de trabajo de ModelArts en un flujo de trabajo.

Escenario de aplicación	Servicio dependiente	Política dependiente	Función admitida
Gestión de datos	OBS	obs:bucket:ListAllMybuckets (Obtaining a bucket list) obs:bucket:HeadBucket (Obtaining bucket metadata) obs:bucket:ListBucket (Listing objects in a bucket) obs:bucket:GetBucketLocation (Obtaining the bucket location) obs:object:GetObject (Obtaining object content and metadata) obs:object:GetObjectVersion (Obtaining object content and metadata) obs:object:PutObject (Uploading objects using PUT method, uploading objects using POST method, copying objects, appending an object, initializing a multipart task, uploading parts, and merging parts) obs:object:DeleteObject (Deleting an object or batch deleting objects) obs:object:DeleteObjectVersion (Deleting an object or batch deleting objects) obs:object:ListMultipartUploadParts (Listing uploaded parts) obs:object:AbortMultipartUpload (Aborting multipart uploads) obs:object:GetObjectAcl (Obtaining an object ACL) obs:object:GetObjectVersionAcl (Obtaining an object ACL) obs:bucket:PutBucketAcl (Configuring a bucket ACL) obs:object:PutObjectAcl (Configuring an object ACL)	Utilizar datos de OBS en un flujo de trabajo.
Ejecución de un flujo de trabajo	IAM	iam:users:listUsers (Obtaining users) iam:agencies:getAgency (Obtaining details about a specified agency) iam:tokens:assume (Obtaining an agency token)	Llame a otros servicios de ModelArts cuando se ejecute el flujo de trabajo.

Escenario de aplicación	Servicio dependiente	Política dependiente	Función admitida
Integración de DLI	DLI	dli:jobs:get (Obtaining job details) dli:jobs:list_all (Viewing a job list) dli:jobs:create (Creating a job)	Integrar DLI en un flujo de trabajo.
Integración de MRS	MRS	mrs:job:get (Obtaining job details) mrs:job:submit (Creating and executing a job) mrs:job:list (Viewing a job list) mrs:job:stop (Stopping a job) mrs:job:batchDelete (Batch deleting jobs) mrs:file:list (Viewing a file list)	Integrar MRS en un flujo de trabajo.

Tabla 2-11 Gestión de aplicaciones de IA

Escenario de aplicación	Servicio dependiente	Política dependiente	Función admitida
Gestión de aplicaciones de IA	SWR	swr:repository:deleteRepository swr:repository:deleteTag swr:repository:getRepository swr:repository:listTags	Importe un modelo desde una imagen personalizada. Utilice un motor personalizado al importar un modelo de OBS.

Escenario de aplicación	Servicio dependiente	Política dependiente	Función admitida
	OBS	obs:bucket:ListAllMybuckets (Obtaining a bucket list) obs:bucket:HeadBucket (Obtaining bucket metadata) obs:bucket:ListBucket (Listing objects in a bucket) obs:bucket:GetBucketLocation (Obtaining the bucket location) obs:object:GetObject (Obtaining object content and metadata) obs:object:GetObjectVersion (Obtaining object content and metadata) obs:object:PutObject (Uploading objects using PUT method, uploading objects using POST method, copying objects, appending an object, initializing a multipart task, uploading parts, and merging parts) obs:object:DeleteObject (Deleting an object or batch deleting objects) obs:object:DeleteObjectVersion (Deleting an object or batch deleting objects) obs:object:ListMultipartUploadParts (Listing uploaded parts) obs:object:AbortMultipartUpload (Aborting multipart uploads) obs:object:GetObjectAcl (Obtaining an object ACL) obs:object:GetObjectVersionAcl (Obtaining an object ACL) obs:bucket:PutBucketAcl (Configuring a bucket ACL) obs:object:PutObjectAcl (Configuring an object ACL)	Importe un modelo desde una plantilla. Especificar una ruta de OBS para la conversión del modelo.

Tabla 2-12 Gestión de despliegue de servicio

Escenario de aplicación	Servicio dependiente	Política dependiente	Función admitida
Servicios en tiempo real	LTS	lts:logs:list (Obtaining the log list)	Mostrar logs de LTS.
	OBS	obs:bucket:GetBucketPolicy (Obtaining a bucket policy) obs:bucket:HeadBucket (Obtaining bucket metadata) obs:bucket:ListAllMyBuckets (Obtaining a bucket list) obs:bucket:PutBucketPolicy (Configuring a bucket policy) obs:bucket:DeleteBucketPolicy (Deleting a bucket policy)	Montar volúmenes externos en un contenedor cuando se ejecuten los servicios.
Servicios por lotes	OBS	obs:object:GetObject (Obtaining object content and metadata) obs:object:PutObject (Uploading objects using PUT method, uploading objects using POST method, copying objects, appending an object, initializing a multipart task, uploading parts, and merging parts) obs:bucket:CreateBucket (Creating a bucket) obs:bucket:ListBucket (Listing objects in a bucket) obs:bucket:ListAllMyBuckets (Obtaining a bucket list)	Crear servicios por lotes y realizar inferencias por lotes.
Servicios de borde	CES	ces:metricData:list: (Obtaining metric data)	Consultar las métricas de monitoreo.
	IEF	ief:deployment:delete (Deleting a deployment)	Gestionar servicios perimetrales.

Tabla 2-13 Gestión de conjuntos de datos

Escenario de aplicación	Servicio dependiente	Política dependiente	Función admitida
Gestión de conjuntos de datos y etiquetas	OBS	obs:bucket:ListBucket (Listing objects in a bucket) obs:object:GetObject (Obtaining object content and metadata) obs:object:PutObject (Uploading objects using PUT method, uploading objects using POST method, copying objects, appending an object, initializing a multipart task, uploading parts, and merging parts) obs:object:DeleteObject (Deleting an object or batch deleting objects) obs:bucket:HeadBucket (Obtaining bucket metadata) obs:bucket:GetBucketAcl (Obtaining a bucket ACL) obs:bucket:PutBucketAcl (Configuring a bucket ACL) obs:bucket:GetBucketPolicy (Obtaining a bucket policy) obs:bucket:PutBucketPolicy (Configuring a bucket policy) obs:bucket:DeleteBucketPolicy (Deleting a bucket policy) obs:bucket:PutBucketCORS (Configuring or deleting CORS rules of a bucket) obs:bucket:GetBucketCORS (Obtaining the CORS rules of a bucket) obs:object:PutObjectAcl (Configuring an object ACL)	Gestionar datasets en OBS. Etiquetar datos de OBS. Crear un trabajo de gestión de datos.
Gestión de conjuntos de datos de tablas	DLI	dli:database:displayAllDatabases dli:database:displayAllTables dli:table:describe_table	Gestión de datos de DLI en un conjunto de datos.
Gestión de conjuntos de datos de tablas	DWS	dws:openAPICluster:list dws:openAPICluster:getDetail	Gestionar datos de DWS en un conjunto de datos.

Escenario de aplicación	Servicio dependiente	Política dependiente	Función admitida
Gestión de conjuntos de datos de tablas	MRS	mrs:job:submit mrs:job:list mrs:cluster:list mrs:cluster:get	Gestionar datos de MRS en un conjunto de datos.
Etiquetado automático	ModelArts	modelarts:service:list modelarts:model:list modelarts:model:get modelarts:model:create modelarts:trainJobInnerModel:list modelarts:workspace:get modelarts:workspace:list	Habilitar etiquetado automático.
Etiquetado en equipo	IAM	iam:projects:listProjects (Obtaining tenant projects) iam:users:listUsers (Obtaining users) iam:agencies:createAgency (Creating an agency) iam:quotas:listQuotasForProject (Obtaining the quotas of a project)	Gestionar equipos de etiquetado.

Tabla 2-14 Gestión de recursos

Escenario de aplicación	Servicio dependiente	Política dependiente	Función admitida
Gestión de grupos de recursos	BSS	bss:coupon:view bss:order:view bss:balance:view bss:discount:view bss:renewal:view bss:bill:view bss:contract:update bss:order:pay bss:unsubscribe:update bss:renewal:update bss:order:update	Crear, renovar y cancelar la suscripción de un grupo de recursos. Los permisos dependientes debe configurarse en la vista del proyecto de IAM.

Escenario de aplicación	Servicio dependiente	Política dependiente	Función admitida
	ECS	ecs:availabilityZones:list	Mostrar las AZ. Los permisos dependientes debe configurarse en la vista del proyecto de IAM.
Gestión de red	VPC	vpc:routes:create vpc:routes:list vpc:routes:get vpc:routes:delete vpc:peerings:create vpc:peerings:accept vpc:peerings:get vpc:peerings:delete vpc:routeTables:update vpc:routeTables:get vpc:routeTables:list vpc:vpcs:create vpc:vpcs:list vpc:vpcs:get vpc:vpcs:delete vpc:subnets:create vpc:subnets:get vpc:subnets:delete vpcep:endpoints:list vpcep:endpoints:create vpcep:endpoints:delete vpcep:endpoints:get vpc:ports:create vpc:ports:get vpc:ports:update vpc:ports:delete vpc:networks:create vpc:networks:get vpc:networks:update vpc:networks:delete	Crear y eliminar redes de ModelArts e interconectar las VPC. Los permisos dependientes debe configurarse en la vista del proyecto de IAM.

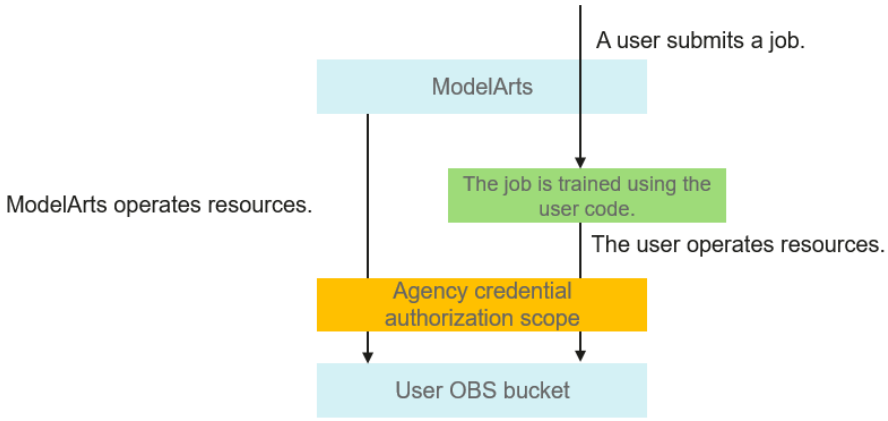
Escenario de aplicación	Servicio dependiente	Política dependiente	Función admitida
	SFS Turbo	sfsturbo:shares:addShareNic sfsturbo:shares:deleteShareNic sfsturbo:shares:showShareNic sfsturbo:shares:listShareNics	Interconecte su red con SFS Turbo. Los permisos dependientes debe configurarse en la vista del proyecto de IAM.
Grupo de recursos de borde	IEF	ief:node:list ief:group:get ief:application:list ief:application:get ief:node:listNodeCert ief:node:get ief:IEFInstance:get ief:deployment:list ief:group:listGroupInstanceState ief:IEFInstance:list ief:deployment:get ief:group:list	Agregar, eliminar, modificar y buscar grupos de borde

Autorización de delegación

Para simplificar las operaciones cuando se utiliza ModelArts para ejecutar trabajos, ciertas operaciones se realizan automáticamente en el backend ModelArts. Por ejemplo, descargar los conjuntos de datos de un bucket de OBS a un espacio de trabajo antes de iniciar un trabajo de entrenamiento y volcar los logs del trabajo de entrenamiento al bucket de OBS.

ModelArts no guarda sus credenciales de autenticación de token. Antes de realizar operaciones en sus recursos (como buckets de OBS) en un trabajo asincrono de backend, debe autorizar explícitamente ModelArts con una delegación de IAM. ModelArts utilizará la delegación para obtener una credencial de autenticación temporal para realizar operaciones con sus recursos. Para obtener más detalles, consulte **Agregación de autorización**.

Figura 2-4 Autorización de delegación



Como se muestra en **Figura 2-4**, una vez configurada la autorización en ModelArts, ModelArts utiliza la credencial temporal para acceder a sus recursos y utilizarlos, liberándole de algunas operaciones complejas y lentas. La credencial de la delegación también se sincronizará con sus trabajos (incluidas las instancias de notebook y los trabajos de entrenamiento). Puede utilizar la credencial de la delegación para acceder a sus recursos en los trabajos.

Puede utilizar cualquiera de los siguientes métodos para autorizar ModelArts utilizando una delegación:

Autorización con un clic

ModelArts proporciona la autorización automática con un solo clic. En la página **Global Configuration** de ModelArts puede configurar rápidamente la autorización de la delegación. A continuación, ModelArts creará automáticamente una delegación para usted y la configurará en ModelArts.

En este modo, el ámbito de autorización se especifica en función de las políticas del sistema preestablecidas de los servicios dependientes para garantizar permisos suficientes para utilizar los servicios. La delegación creada cuenta con casi todos los permisos de los servicios dependientes. Si desea controlar con precisión el alcance de los permisos concedidos a una delegación, utilice el segundo método.

Autorización personalizada

The administrator creates different agency authorization policies for different users in IAM, and configures the created agency for ModelArts users. Cuando se crea una agencia para un usuario de IAM, el administrador especifica los permisos mínimos para la delegación basándose en los permisos del usuario para controlar los recursos a los que el usuario puede acceder cuando utiliza ModelArts. Para obtener más información, consulte **Asignación de permisos básicos para utilizar ModelArts**.

Riesgos de las operaciones no autorizadas

La autorización de delegación de un usuario es independiente. En teoría, el alcance de autorización de delegación de un usuario puede exceder el alcance de autorización de la política de autorización configurada para el grupo de usuarios. Cualquier configuración incorrecta dará lugar a operaciones no autorizadas.

Para evitar operaciones no autorizadas, solo un administrador de tenant puede configurar agencias para usuarios en la configuración global de ModelArts para garantizar la seguridad de la autorización de agencias.

Autorización mínima de delegación

Al configurar la autorización de la delegación, el administrador debe controlar estrictamente el alcance de la autorización.

ModelArts realiza de forma asincrona y automática operaciones como la preparación y la limpieza de trabajos. La autorización requerida de la agencia está dentro del ámbito de la autorización básica. Si utiliza solo algunas funciones de ModelArts, el administrador puede filtrar los permisos básicos que no se utilizan de acuerdo con la configuración de autorización de la delegación. Por el contrario, si necesita obtener permisos de recursos más allá del ámbito de autorización básico en un trabajo, el administrador puede agregar nuevos permisos a la configuración de autorización de la agencia. En una palabra, el alcance de la autorización de agencias debe minimizarse y personalizarse según los requerimientos del servicio.

Ámbito básico de la autorización de delegación

Para personalizar el permiso para una delegación, seleccione los permisos en función de sus requisitos de servicio.

Tabla 2-15 Autorización básica de delegación para un entorno de desarrollo

Escenario de aplicación	Servicio dependiente	Autorización de delegación	Descripción	Sugerencia de configuración
JupyterLab	OBS	obs:object:DeleteObject obs:object:GetObject obs:object:GetObjectVersion obs:bucket:CreateBucket obs:bucket:ListBucket obs:bucket:ListAllMyBuckets obs:object:PutObject obs:bucket:GetBucketAcl obs:bucket:PutBucketAcl obs:bucket:PutBucketCORS	Utilice OBS para cargar y descargar datos en JupyterLab con notebook de ModelArts.	Recomendado

Escenario de aplicación	Servicio dependiente	Autorización de delegación	Descripción	Sugerencia de configuración
Monitoreo del entorno de desarrollo	AOM	aom:alarm:put	Invoque a la API de AOM para obtener datos de monitoreo y eventos de instancias de notebook y mostrarlos en el notebook de ModelArts.	Recomendado

Tabla 2-16 Autorización básica de delegación para trabajos de entrenamiento

Escenario de aplicación	Servicio dependiente	Autorización de delegación	Descripción
Trabajos de entrenamiento	OBS	obs:bucket:ListBucket obs:object:GetObject obs:object:PutObject	Descargue datos, modelos y código antes de comenzar un trabajo de entrenamiento. Cargue logs y modelos cuando se ejecute un trabajo de entrenamiento.

Tabla 2-17 Autorización básica de delegación para desplegar servicios

Escenario de aplicación	Servicio dependiente	Autorización de delegación	Descripción
Servicios en tiempo real	LTS	lts:groups:create lts:groups:list lts:topics:create lts:topics:delete lts:topics:list	Configure LTS para reportar logs de servicios en tiempo real.

Escenario de aplicación	Servicio dependiente	Autorización de delegación	Descripción
Servicios por lotes	OBS	obs:bucket:ListBucket obs:object:GetObject obs:object:PutObject	Ejecutar un servicio por lotes.
Servicios de borde	IEF	ief:deployment:list ief:deployment:create ief:deployment:update ief:deployment:delete ief:node:createNodeCert ief:iefInstance:list ief:node:list	Despliegue un servicio de borde con IEF.

Tabla 2-18 Autorización básica de delegación para gestionar datos

Escenario de aplicación	Servicio dependiente	Autorización de delegación	Descripción
Conjunto de datos y etiquetado de datos	OBS	obs:object:GetObject obs:object:PutObject obs:object:DeleteObject obs:object:PutObjectAcl obs:bucket:ListBucket obs:bucket:HeadBucket obs:bucket:GetBucketAcl obs:bucket:PutBucketAcl obs:bucket:GetBucketPolicy obs:bucket:PutBucketPolicy obs:bucket:DeleteBucketPolicy obs:bucket:PutBucketCORS obs:bucket:GetBucketCORS	Gestionar conjuntos de datos en un bucket de OBS.
Etiquetado de datos	ModelArts inference	modelarts:service:get modelarts:service:create modelarts:service:update	Realizar el etiquetado automático basado en la inferencia de ModelArts.

Tabla 2-19 Autorización básica de delegación para gestionar grupos de recursos dedicados

Escenario de aplicación	Servicio dependiente	Autorización de delegación	Descripción
Gestión de redes (nueva versión)	VPC	vpc:routes:create vpc:routes:list vpc:routes:get vpc:routes:delete vpc:peerings:create vpc:peerings:accept vpc:peerings:get vpc:peerings:delete vpc:routeTables:update vpc:routeTables:get vpc:routeTables:list vpc:vpcs:create vpc:vpcs:list vpc:vpcs:get vpc:vpcs:delete vpc:subnets:create vpc:subnets:get vpc:subnets:delete vpcep:endpoints:list vpcep:endpoints:create vpcep:endpoints:delete vpcep:endpoints:get vpc:ports:create vpc:ports:get vpc:ports:update vpc:ports:delete vpc:networks:create vpc:networks:get vpc:networks:update vpc:networks:delete	Crear y eliminar redes de ModelArts e interconectar las VPC. Los permisos dependientes debe configurarse en la vista del proyecto de IAM.

Escenario de aplicación	Servicio dependiente	Autorización de delegación	Descripción
	SFS Turbo	sfsturbo:shares:addShareNic sfsturbo:shares:deleteShareNic sfsturbo:shares:showShareNic sfsturbo:shares:listShareNics	Interconecte su red con SFS Turbo. Los permisos dependientes debe configurarse en la vista del proyecto de IAM.
Gestión de grupos de recursos	BSS	bss:coupon:view bss:order:view bss:balance:view bss:discount:view bss:renewal:view bss:bill:view bss:contract:update bss:order:pay bss:unsubscribe:update bss:renewal:update bss:order:update	Crear, renovar y cancelar la suscripción de un grupo de recursos. Los permisos dependientes debe configurarse en la vista del proyecto de IAM.
Gestión de grupos de recursos	ECS	ecs:availabilityZones:list	Mostrar las AZ. Los permisos dependientes debe configurarse en la vista del proyecto de IAM.

2.2.3 Espacio de trabajo

ModelArts le permite crear múltiples espacios de trabajo para desarrollar algoritmos y gestionar y desplegar modelos para diferentes objetivos de servicio. De este modo, los productos de desarrollo de las distintas aplicaciones se asignan a diferentes espacios de trabajo para simplificar la gestión.

Espacio de trabajo soporta los siguientes tipos de control de acceso:

- **PUBLIC**: acceso público para los tenants (incluidas las cuentas de tenant y todas sus cuentas de usuario)
- **PRIVATE**: solo puede acceder a las cuentas de creador y de tenant
- **INTERNAL**: accesible para el creador, las cuentas de tenant y las cuentas de usuario de IAM especificadas. Cuando **Authorization Type** se establece en **INTERNAL**, especifique una o más cuentas de usuario de IAM accesibles.

Se asigna un espacio de trabajo por defecto a cada proyecto IAM de cada cuenta. El control de acceso del espacio de trabajo predeterminado es **PUBLIC**.

El control de acceso al espacio de trabajo permite el acceso solo a determinados usuarios. Esta función se puede utilizar en los siguientes escenarios:

- **Education**: Un profesor asigna un espacio de trabajo **INTERNAL** a cada alumno y permite que solo los alumnos especificados puedan acceder a los espacios de trabajo. De este modo, los alumnos pueden realizar experimentos por separado en ModelArts.
- **Enterprises**: Un administrador crea un espacio de trabajo para tareas de producción y permite que solo el personal de O&M utilice el espacio de trabajo, y crea un espacio de trabajo para depuración rutinaria y permite que solo los desarrolladores utilicen el espacio de trabajo. De esta manera, los diferentes roles de empresa pueden usar recursos solo en un espacio de trabajo especificado.

Como usuario empresarial, puede enviar la solicitud para habilitar la función de espacio de trabajo a su soporte técnico.

2.3 Prácticas de configuración en escenarios típicos

2.3.1 Asignación de permisos a los usuarios individuales para utilizar ModelArts

Ciertas funciones de ModelArts requieren el acceso al Object Storage Service (OBS), al Software Repository for Container (SWR) y al Intelligent EdgeFabric (IEF). Antes de utilizar ModelArts, su cuenta debe estar autorizada para acceder a estos servicios. De lo contrario, estas funciones no estarán disponibles.

Restricciones

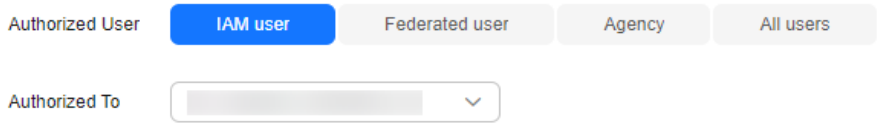
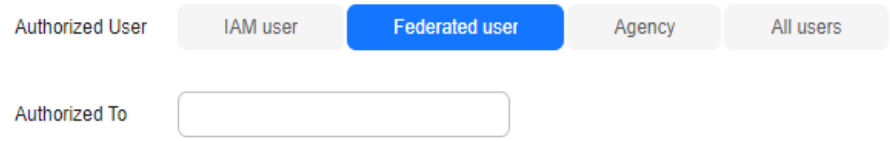
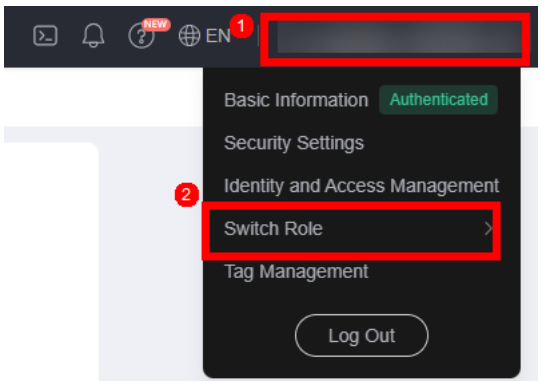
- Solo una cuenta de tenant puede realizar la autorización de la delegación para autorizar la cuenta actual o a todos los usuarios de IAM con la cuenta actual.
- Varios usuarios o cuentas de IAM pueden usar la misma delegación.
- Se puede crear un máximo de 50 delegaciones bajo una cuenta.
- Si utiliza ModelArts por primera vez, agregue una delegación. En general, los permisos de usuario comunes son suficientes para satisfacer sus necesidades. Puede configurar los permisos para una gestión refinada de los mismos.
- Si no ha sido autorizado, el ModelArts mostrará un mensaje indicando que no ha sido autorizado al acceder a la página del **Add Authorization**. En este caso, póngase en contacto con el administrador para agregar autorización.

Adición de autorización

1. Inicie sesión en la consola de gestión de ModelArts. En el panel de navegación de la izquierda, seleccione **Settings**. Se muestra la página **Global Configuration**.
2. Haga clic en **Add Authorization**. En la página **Add Authorization** que se muestra, configure los parámetros.

Tabla 2-20 Parámetros

Parámetro	Descripción
Authorized User	Opciones: **IAM user**, **Federated user**, **Agency** y **All users** ● **IAM user**: Puede utilizar una cuenta de tenant para crear usuarios de IAM y asignar permisos para recursos específicos. Cada usuario de IAM tiene sus propias credenciales de identidad (contraseña y claves de acceso) y utiliza recursos en la nube basados en los permisos asignados. Para obtener más información sobre los usuarios de IAM, consulte **Usuario de IAM**. ● **Federated user**: un usuario federado también se denomina usuario empresarial virtual. Para obtener más información sobre usuarios federados, consulte **Configuración de autenticación de identidades federadas**. ● **Agency**: Puede crear delegación en IAM. Para obtener más información sobre cómo crear una delegación, consulte **Creación de una delegación**. ● **All users**: Si selecciona esta opción, los permisos de delegación se otorgará a todos los usuarios de IAM con la cuenta actual, incluidos los creados en el futuro. Seleccione **All users** para los usuarios individuales.

Parámetro	Descripción
Authorized To	Este parámetro no se muestra cuando **Authorized User** se establece en **All users**. ● **IAM user**: seleccione un usuario de IAM y configure una delegación para el usuario de IAM. **Figura 2-5** Selección de un usuario de IAM  ● **Federated user**: Introduzca el nombre de usuario o ID de usuario del usuario federado de destino. **Figura 2-6** Selección de un usuario federado  ● **Agency**: seleccione el nombre de delegación. Puede crear una delegación en la cuenta A y conceder los permisos de la agencia a la cuenta B. Al usar la cuenta B, puede cambiar el rol en la esquina superior derecha de la consola a la cuenta A y usar los permisos de la delegación de la cuenta A. **Figura 2-7** Cambio del rol 
Agency	● **Use existing**: Si hay agencias en la lista, seleccione una disponible para autorizar al usuario seleccionado. Haga clic en la flecha desplegable junto al nombre de una delegación para ver los detalles de su permiso. ● **Add agency**: si no hay delegación disponible, cree una. Si usa ModelArts por primera vez, seleccione **Add agency**.
Add agency > Agency Name	El sistema crea automáticamente un nombre de delegación cambiable.

Parámetro	Descripción
Add agency > Authorization Method	● **Role-based**: Una estrategia de autorización de IAM de grano grueso para asignar permisos en función de las responsabilidades del usuario. Solo hay disponible un número limitado de roles de nivel de servicio. Cuando utilice los roles para otorgar permisos, asigne otros roles de los que dependan los permisos para que surtan efecto. Los roles no son ideales para la autorización detallada y el control de acceso seguro. ● **Policy-based**: Herramienta de autorización detallada que define permisos para operaciones en recursos específicos de la nube en determinadas condiciones. Este tipo de autorización es más flexible e ideal para un control de acceso seguro. Para obtener más detalles sobre funciones y políticas, consulte **Conceptos básicos**.
Add agency > Permissions > Common User	**Common User** proporciona los permisos para usar todas las funciones básicas de ModelArts. Por ejemplo, puede acceder a los datos y crear y gestionar trabajos de entrenamiento. Seleccione esta opción en general. Haga clic en **View permissions** para ver los permisos de usuario comunes.
Add agency > Permissions > Custom	Si necesita una gestión de permisos mejorada, seleccione **Custom** para asignar permisos de forma flexible a la delegación creada. Puede seleccionar permisos de la lista de permisos según sea necesario.

3. Seleccione **I have read and agree to the ModelArts Service Statement**. Haga clic en **Create**.

Consulta de permisos autorizados

Puede ver las autorizaciones configuradas en la página **Global Configuration**. Haga clic en **View Permissions** en la columna **Authorization Content** para ver los detalles del permiso.

Figura 2-8 Consulta de permisos

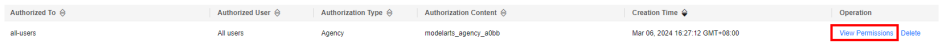
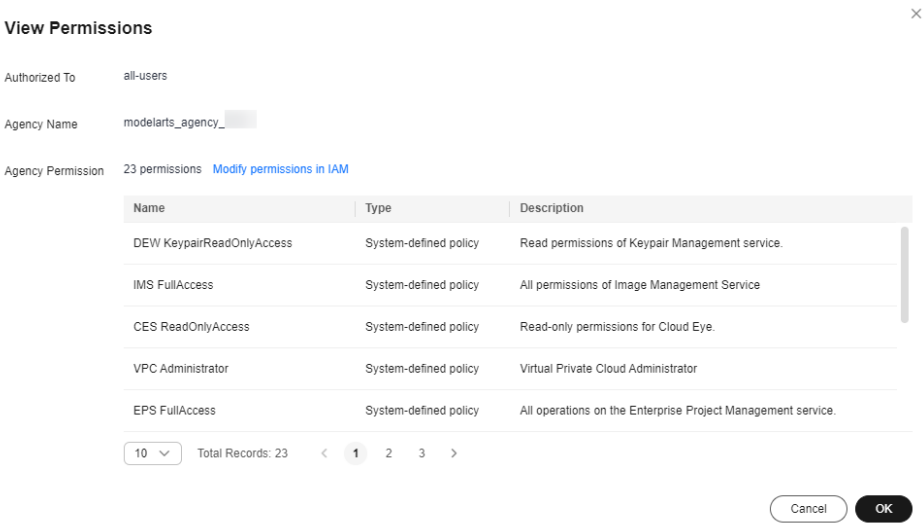


Figura 2-9 Permisos comunes de usuario



2.3.2 Asignación de permisos básicos para utilizar ModelArts

2.3.2.1 Escenarios

Algunas funciones de ModelArts requieren permiso para acceder a otros servicios. Esta sección describe cómo asignar permisos específicos a los usuarios de IAM cuando utilizan ModelArts.

Permisos

Los permisos de los usuarios de IAM son controlados por el usuario de tenant. Al iniciar sesión como usuario de tenant, puede asignar permisos al grupo de usuarios objetivo con IAM. Luego, los permisos se asignan a todos los miembros del grupo de usuarios. La siguiente lista de autorizaciones utiliza como ejemplo las políticas definidas por el sistema de ModelArts y otros servicios.

Tabla 2-21 Autorización de servicio

Servicio de destino	Descripción	Permisos de IAM	Obligatorio
ModelArts	Asignar los permisos a los usuarios de IAM para usar ModelArts. Los usuarios con permiso de ModelArts CommonOperations solo pueden usar recursos, pero no pueden crear, actualizar ni eliminar ningún grupo de recursos dedicado. Se recomienda asignar este permiso a los usuarios de IAM.	ModelArts CommonOperations	Sí
	Los usuarios con permiso de ModelArts FullAccess tienen los permisos de acceso completos, incluidas creación, actualización y eliminación de los grupos de recursos dedicados. Tenga cuidado al seleccionar esta opción.	ModelArts FullAccess	No Seleccione **ModelArts FullAccess** o **ModelArts CommonOperations**.

Servicio de destino	Descripción	Permisos de IAM	Obligatorio
Object Storage Service (OBS)	Asignar permisos a los usuarios de IAM para usar OBS. Todas las operaciones de ModelArts, como la gestión de datos, los entornos de desarrollo, los trabajos de entrenamiento y los despliegue de modelo, necesitan OBS para el reenvío de datos.	OBS OperateAccess	Sí
Software Repository for Container (SWR)	Asignar permisos a los usuarios de IAM para usar SWR. Las imágenes personalizadas de ModelArts requieren el permiso del SWR FullAccess.	SWR OperateAccess	Sí
Key Management Service (KMS)	Para utilizar el SSH remoto del notebook de ModelArts, los usuarios de IAM requieren la autorización de KMS.	KMS CMKFullAccess	No
Intelligent EdgeFabric (IEF)	Asignar permisos a los usuarios de IAM para usar IEF. Se requiere los permisos de administrador de tenant para que se puedan utilizar los servicios edge de ModelArts que dependen de IEF.	Tenant Administrator	No
Cloud Eye	Asignar permisos a los usuarios de IAM para usar Cloud Eye. Con Cloud Eye, puede ver los estados de ejecución de los servicios en tiempo real de ModelArts y las cargas de aplicaciones de AI, y establecer alarmas de supervisión.	CES FullAccess	No
Simple Message Notification (SMN)	Asignar permisos a los usuarios de IAM para usar SMN. SMN se utiliza con Cloud Eye.	SMN FullAccess	No

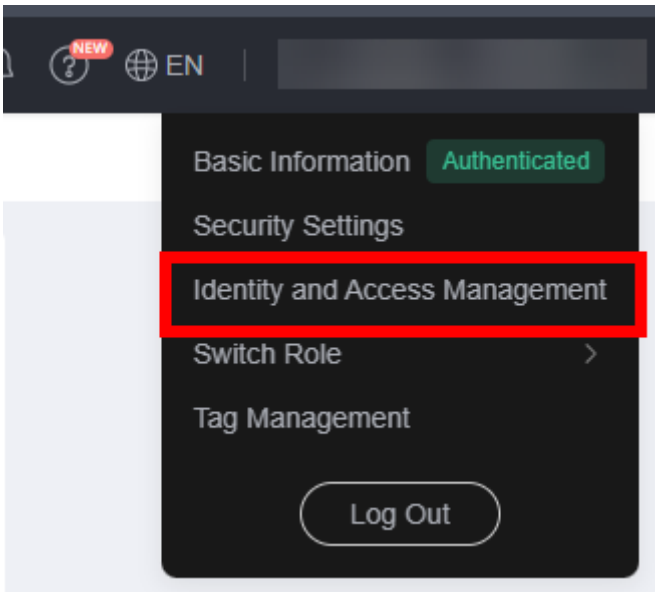
Servicio de destino	Descripción	Permisos de IAM	Obligatorio
Virtual Private Cloud (VPC)	Durante la creación de un grupo de recursos dedicado para ModelArts, los usuarios de IAM requieren permisos de VPC para poder personalizar las redes.	VPC FullAccess	No
Scalable File Service (SFS)	Asignar permisos a los usuarios de IAM para usar SFS. Los file systems de SFS se pueden montar en grupos de recursos dedicados de ModelArts para servir de almacenamiento para entornos de desarrollo o entrenamiento.	SFS Turbo FullAccess SFS FullAccess	No

2.3.2.2 Paso 1 Cree un grupo de usuarios y agregue datos al grupo de usuarios

Se pueden crear varios usuarios de IAM bajo un usuario tenant y se gestionan los permisos de los usuarios de IAM por grupo. Esta sección describe cómo crear un grupo de usuarios y los usuarios de IAM y agregarlos a ese grupo.

1. Inicie sesión en la consola de gestión como usuario tenant, pase el mouse por encima de su nombre de usuario en la esquina superior derecha y elija **Identity and Access Management** en la lista desplegable para pasar a la consola de gestión de IAM.

Figura 2-10 Gestión de identidades y accesos



2. Cree un grupo de usuarios. En el panel de navegación de la izquierda, elija **User Groups**. Haga clic en **Create User Group** en la esquina superior derecha. Luego, configure **Name** como **UserGroup-2** y haga clic en **OK**.

   Una vez creado el grupo de usuarios, el sistema cambia automáticamente a la lista de grupos de usuarios. Luego, puede agregar los usuarios de IAM existentes al grupo de usuarios con la gestión de grupos de usuarios. Si no existe ningún usuario de IAM, créelo y agréguelo al grupo de usuarios.

3. Cree usuarios de IAM y agréguelos al grupo de usuarios. En el panel de navegación de la izquierda, seleccione **Users**. En la página que aparece, haga clic en **Create User** en la esquina superior derecha. En la página **Create User**, agregue varios usuarios.

   Defina los parámetros según se le solicite y haga clic en **Next**.

4. En la página **Add User to Group**, seleccione **UserGroup-2** y haga clic en **Create**.

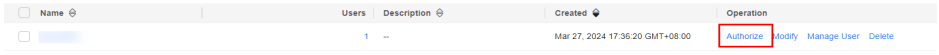
   El sistema agregará automáticamente los dos usuarios al grupo objetivo uno por uno.

2.3.2.3 Paso 2 Asignar permisos para el uso de servicios en la nube

Un usuario de IAM puede utilizar servicios en la nube como ModelArts y OBS solo después de que el usuario de tenant le asigne los permisos. Esta sección describe cómo asignar los permisos para utilizar servicios en la nube a todos los usuarios de IAM de un grupo de usuarios.

1. En la página de lista de grupos de usuarios de IAM, haga clic en **Authorize** del grupo de usuarios de destino. Aparecerá la página **Authorize User Group**.

   Figura 2-11 Autorización

2. Antes de asignar permisos, conozca los requisitos mínimos de permisos de cada módulo de ModelArts, como se muestra en **Tabla 2-21**.

3. Asigne los permisos para usar ModelArts. Busque ModelArts en el cuadro de búsqueda. Seleccione **ModelArts FullAccess** o **ModelArts CommonOperations**.

   Las diferencias entre las opciones son las siguientes:

   – Los usuarios con permiso de ModelArts CommonOperations solo pueden usar recursos, pero no pueden crear, actualizar ni eliminar ningún grupo de recursos dedicado. Se recomienda asignar este permiso a los usuarios de IAM.

   – Los usuarios con permiso de ModelArts FullAccess tienen los permisos de acceso completos, incluidas creación, actualización y eliminación de los grupos de recursos dedicados. Tenga cuidado al seleccionar esta opción.

4. Asigne los permisos para el uso de OBS. Busque **OBS** y seleccione **OBS Administrator**. Los trabajos de entrenamiento de ModelArts utilizan OBS para reenviar datos. Por lo tanto, son necesarios los permisos para utilizar OBS.

5. Asigne los permisos para el uso de SWR. Busque **SWR** y seleccione **SWR FullAccess**. Las imágenes personalizadas de ModelArts requieren el permiso del SWR FullAccess.

6. (Opcional) Asigne el permiso de gestión de claves. El SSH remoto del notebook de ModelArts requiere permiso de gestión de claves. Busque **DEW** y seleccione **DEW KeypairFullAccess**.

   El permiso de gestión de claves de DEW se configura en las siguientes regiones: **CN North-Beijing1**, **CN North-Beijing4**, **CN East-Shanghai1**, **CN East-Shanghai2**, **CN**

South-Guangzhou, **CN Southwest-Guiyang1**, **CN-Hong Kong** y **AP-Singapore**. En otras regiones, se configura el permiso de gestión de claves de KMS. En este ejemplo, se utiliza la región **CN-Hong Kong**. Por lo tanto, se debe configurar el permiso de gestión de claves de DEW.

7. (Opcional) Asigne los permisos para utilizar IEF. ModelArts requiere el permiso del Tenant Administrator para poder utilizar los servicios perimetrales que dependen del IEF.

   Tenant Administrator tiene permiso para gestionar todos los servicios en la nube, no solo ModelArts. Tenga cuidado al asignar el permiso de Tenant Administrator.

8. (Opcional) Asigne permisos para utilizar Cloud Eye y SMN. En la página de detalles de un servicio en tiempo real de ModelArts desplegado para inferencia, el número de llamadas está disponible. Haga clic en **View Details** para obtener más información. Si desea ver el estado general de ejecución de los servicios en tiempo real de ModelArts y las cargas de aplicaciones de AI en Cloud Eye, asigne permisos de Cloud Eye a los usuarios de IAM.

   Seleccione **CES ReadOnlyAccess** para ver solo los datos de monitoreo.

   Para configurar el monitoreo de alarmas en Cloud Eye, también debe agregar **CES FullAccess** y SMN.

9. (Opcional) Asigne permisos para utilizar VPC. Para habilitar la configuración de red personalizada al crear un grupo de recursos dedicado, asigne permisos para usar VPC.
10. (Opcional) Asigne permisos para usar SFS y SFS Turbo. Para montar un sistema de SFS en un grupo de recursos dedicado como almacenamiento para el entorno de desarrollo o entrenamiento, asigne el permiso para usar el sistema de SFS.
11. Haga clic en **View Selected** en la esquina superior izquierda y confirme el permiso seleccionado.
12. Haga clic en **Next** y configure el alcance mínimo de autorización. Seleccione **Region-specific projects**, seleccione la región que desea autorizar y haga clic en **OK**.
13. Aparece un mensaje que indica que la autorización se realizó correctamente. Consulte la información de autorización y haga clic en **Finish**. La autorización tarda entre 15 y 30 minutos en surtir efecto.

2.3.2.4 Paso 3 Configurar la autorización de acceso a ModelArts basada en agentes para el usuario

Después de asignar los permisos de IAM, configure la autorización de acceso de ModelArts para los usuarios de IAM en la página de ModelArts para que ModelArts pueda acceder a servicios dependientes como OBS, SWR e IEF.

En la autorización de acceso de ModelArts basada en agentes, solo los usuarios de tenant pueden configurar para sus usuarios de IAM. En este ejemplo, utilice la cuenta de administrador para configurar la autorización de acceso para todos los usuarios.

1. Utilice la cuenta de tenant para iniciar sesión en la consola de gestión de ModelArts. Seleccione su región en la esquina superior izquierda.
2. En el panel de navegación de la izquierda, seleccione **Settings**. Se muestra la página **Global Configuration**.
3. Haga clic en **Add Authorization**. En la página **Add Authorization**, configure **Authorized User** como **All users** y haga clic en **Add agency** para configurar la autorización basada en agencias para todos los usuarios de IAM bajo la cuenta.

- **Common User**: puede utilizar funciones básicas de ModelArts, por ejemplo, acceder a datos y crear y gestionar trabajos de entrenamiento, pero no para gestionar recursos. Seleccione esta opción en general.
- **Custom**: Puede asignar permisos de forma flexible a la agencia creada. Seleccione esta opción para una mejor gestión de los permisos. Puede seleccionar permisos de la lista de permisos según sea necesario.

4. Seleccione **I have read and agree to the ModelArts Service Statement**. Haga clic en **Create**.

2.3.2.5 Paso 4 Verificar los datos de usuario

Los permisos configurados en 4 tardan entre 15 y 30 minutos en surtir efecto. Por lo tanto, espere 30 minutos después de la configuración y luego verifique la configuración.

1. Inicie sesión en la consola de gestión de ModelArts como un IAM de **UserGroup-2**. En la página de inicio de sesión, asegúrese de que **IAM User Login** esté seleccionado.

   Cambie la contraseña según se le indique al iniciar la sesión por primera vez.

2. Revise los permisos de ModelArts.

   a. Seleccione la región de destino en la esquina superior izquierda, que debe ser la misma que la de la configuración de autorización.

   b. En el panel de navegación situado a la izquierda de la consola de gestión de ModelArts, seleccione **DevEnviron** > **Notebook**. Los permisos de ModelArts y la autorización de la delegación se configuran correctamente si no hay mensaje sobre permisos insuficientes.

      Si aparece un mensaje que indica que debe autorizar el acceso, significa que no se ha configurado la autorización de la delegación de ModelArts. En este caso, siga las instrucciones de **Paso 3 Configurar la autorización de acceso a ModelArts basada en agentes para el usuario** para configurar la autorización.

   c. En el panel de navegación situado a la izquierda de la consola de gestión de ModelArts, seleccione **DevEnviron** > **Notebook** y haga clic en **Create**. Si esta operación se realiza correctamente, habrá obtenido los permisos de la operación de ModelArts.

      También puede probar otras funciones, como **Training Management** > **Training Jobs**. Si la operación se realiza correctamente, puede utilizar ModelArts correctamente.

3. Verifique los permisos de OBS.

   a. En la lista de servicios ubicada en la esquina superior izquierda, seleccione OBS. Aparecerá la consola de gestión del OBS.

   b. Haga clic en **Create Bucket** en la esquina superior derecha. Si esta operación se realiza correctamente, habrá obtenido los permisos de la operación de OBS.

4. Verifique los permisos de SWR.

   a. En la lista de servicios situada en la esquina superior izquierda, seleccione SWR. Aparecerá la consola de gestión de SWR.

   b. Si una página de SWR se puede mostrar correctamente, significa que ha obtenido los permisos de la operación de SWR.

5. Verifique otros permisos opcionales.
6. Experimente ModelArts.

2.3.3 Asignación separada de permisos a administradores y desarrolladores

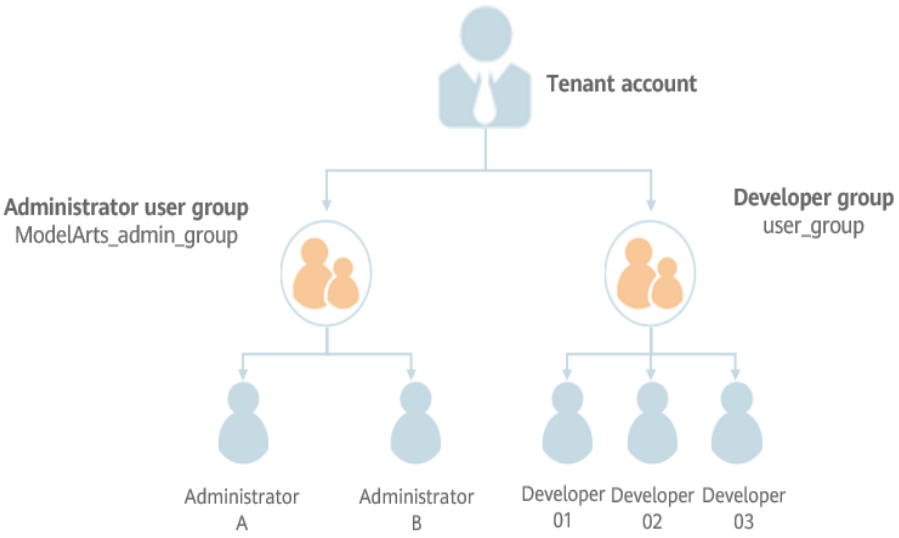
En equipos pequeños y medianos, los administradores deben controlar globalmente los recursos de ModelArts y los desarrolladores solo deben centrarse en sus propias instancias. Por lo general, el permiso **te_admin** de una cuenta de desarrollador debe ser configurado por la cuenta del tenant. Esta sección utiliza el notebook como ejemplo para describir cómo asignar diferentes permisos a administradores y desarrolladores con políticas personalizadas.

Escenarios

Para desarrollar un proyecto con notebook, los administradores necesitan los permisos de control completo para usar grupos de recursos dedicados de ModelArts y los permisos de acceso y de operación en todas las instancias de notebook.

Para usar entornos de desarrollo, los desarrolladores solo necesitan los permisos de operación para usar sus propias instancias y servicios dependientes. No necesitan realizar operaciones en grupos de recursos dedicados de ModelArts ni ver instancias de notebook de otros usuarios.

Figura 2-12 Relaciones entre las cuentas



Configuración de permisos para un administrador

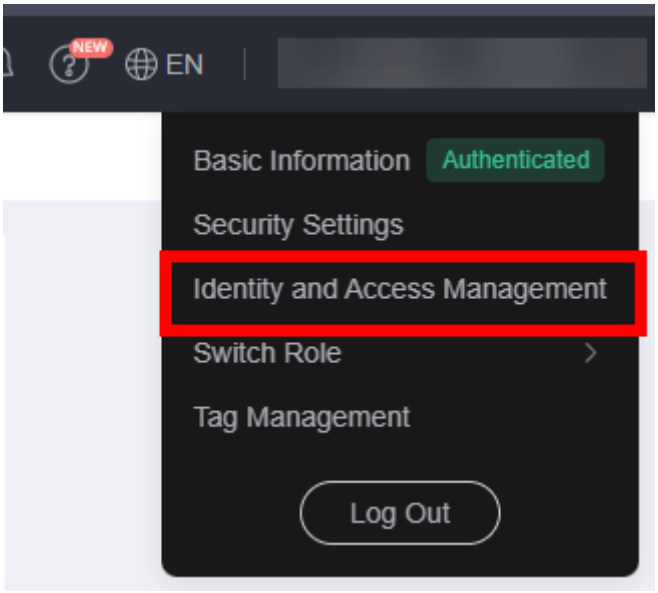
Asigne permisos de control total a los administradores para utilizar los grupos de recursos dedicados de ModelArts y todas las instancias de notebook. El procedimiento es el siguiente:

Paso 1 Utilice una cuenta de tenant para crear un grupo de usuarios de administradores **ModelArts_admin_group** y agregar cuentas de administrador a **ModelArts_admin_group**. Para obtener más información, consulte **Paso 1 Cree un grupo de usuarios y agregue datos al grupo de usuarios**.

Paso 2 Cree una política personalizada.

1. Inicie sesión en la consola de gestión con una cuenta de administrador, coloque el cursor sobre su nombre de usuario en la esquina superior derecha y haga clic en **Identity and Access Management** en la lista desplegable para pasar a la consola de gestión de IAM.

Figura 2-13 Gestión de identidades y accesos



2. Cree la política 1 personalizada y asigne los permisos de IAM y de OBS al usuario. En el panel de navegación de la consola de IAM, seleccione **Permissions** > **Policies/Roles**. Haga clic en **Create Custom Policy** en la esquina superior derecha. En la página mostrada, escriba **Policy1_IAM_OBS** para **Policy Name**, seleccione **JSON** para **Policy View**, configure el contenido de la política y haga clic en **OK**.

   La políticas personalizadas **Policy1_IAM_OBS** es el siguiente, que otorga los permisos de operaciones de IAM y OBS al usuario. Puede copiar y pegar el contenido directamente.

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:users:listUsers",
                "iam:projects:listProjects",
                "obs:object:PutObject",
                "obs:object:GetObject",
                "obs:object:GetObjectVersion",
                "obs:bucket:HeadBucket",
                "obs:object:DeleteObject",
                "obs:bucket:CreateBucket",
                "obs:bucket:ListBucket"
                ]
        }
    ]
}
```

3. Repita **Paso 2.2** para crear la política 2 personalizada y otorgar al usuario el permiso para realizar operaciones en los servicios dependientes ECS, SWR, MRS y SMN, así como ModelArts. Configure **Policy Name** en **Policy2_AllowOperation** y **Policy View** en **JSON**, configure el contenido de la política y haga clic en **OK**.

   La políticas personalizadas **Policy2_AllowOperation** es el siguiente, que otorga al usuario los permisos para realizar operaciones en servicios dependientes ECS, SWR, MRS y SMN como ModelArts. Puede copiar y pegar el contenido directamente.

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Effect": "Allow",
```



```
            "Action": [
                "ecs:serverKeypairs:list",
                "ecs:serverKeypairs:get",
                "ecs:serverKeypairs:delete",
                "ecs:serverKeypairs:create",
                "swr:repository:getNamespace",
                "swr:repository:listNamespaces",
                "swr:repository:deleteTag",
                "swr:repository:getRepository",
                "swr:repository:listTags",
                "swr:instance:createTempCredential",
                "mrs:cluster:get",
                "modelarts:*:*"
            ]
        }
    ]
}
```

Paso 3 Otorgue la política creada en **Paso 2** al grupo de administradores **ModelArts_admin_group**.

1. En el panel de navegación de la consola de IAM, seleccione **User Groups**. En la página **User Groups**, localice la fila que contiene **ModelArts_admin_group**; haga clic en **Authorize** en la columna **Operation** y seleccione **Policy1_IAM_OBS** y **Policy2_AllowOperation**. Haga clic en **Next**.
2. Especifique el ámbito como **All resources** y haga clic en **OK**.

Paso 4 Configure la autorización de acceso a ModelArts basada en agentes para que un administrador permita a ModelArts acceder a los servicios dependientes como OBS.

1. Inicie sesión en la consola de ModelArts con una cuenta de tenant. En el panel de navegación de la izquierda, seleccione **Settings**. Se muestra la página **Global Configuration**.
2. Haga clic en **Add Authorization**. En la página **Add Authorization**, configure **Authorized User** como **IAM user**, seleccione una cuenta de administrador para **Authorized To**, seleccione **Add agency** y seleccione **Common User** para **Permissions**. Los administradores no requieren el control de permisos, por lo que se utiliza la configuración predeterminada **Common User**.
3. Seleccione **I have read and agree to the ModelArts Service Statement**. Haga clic en **Create**.

Paso 5 Pruebe los permisos de administrador.

1. Inicie sesión en la consola de gestión de ModelArts como administrador. En la página de inicio de sesión, asegúrese de que **IAM User Login** esté seleccionado.

   Cambie la contraseña según se le indique al iniciar la sesión por primera vez.
2. En el panel de navegación de la consola de gestión de ModelArts, seleccione **Dedicated Resource Pools** y haga clic en **Create**. Se han asignado los permisos al administrador si la consola no muestra el mensaje de permisos insuficientes.

----Fin

Configuración de permisos para un desarrollador

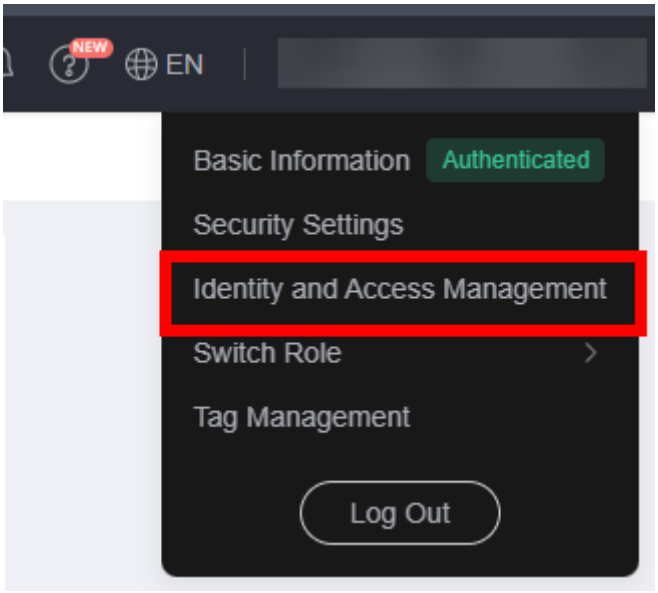
Utilizar IAM para el control de grano fino de los permisos del desarrollador. El procedimiento es el siguiente:

Paso 1 Utilice una cuenta de tenant para crear un grupo de usuarios de desarrolladores **user_group** y agregar cuentas de desarrolladores a **user_group**. Para obtener más información, consulte **Paso 1 Cree un grupo de usuarios y agregue datos al grupo de usuarios**.

Paso 2 Cree una política personalizada.

1. Inicie sesión en la consola de gestión con una cuenta de tenant, coloque el cursor sobre su nombre de usuario en la esquina superior derecha y haga clic en **Identity and Access Management** en la lista desplegable para pasar a la consola de gestión de IAM.

   Figura 2-14 Gestión de identidades y accesos

2. Cree la política 3 personalizada para impedir que los usuarios realicen operaciones en los grupos de recursos dedicados de ModelArts y vean las instancias de notebook de otros usuarios.

   En el panel de navegación de la consola de IAM, seleccione **Permissions** > **Policies/Roles**. Haga clic en **Create Custom Policy** en la esquina superior derecha. En la página mostrada, escriba **Policy3_DenyOperation** para **Policy Name**, seleccione **JSON** para **Policy View**, configure el contenido de la política y haga clic en **OK**.

   La política personalizada **Policy3_DenyOperation** es el siguiente. Puede copiar y pegar el contenido.

```
{
    "Version": "1.1",
    "Statement": [
            {
             "Effect": "deny",
             "Action": [
                 "modelarts:pool:create",
                 "modelarts:pool:update",
                 "modelarts:pool:delete",
            "modelarts:notebook:listAllNotebooks"
                 ]
            }
    ]
}
```

Paso 3 Otorgue la política personalizada al grupo de usuarios del desarrollador **user_group**.

1. En el panel de navegación de la consola de IAM, seleccione **User Groups**. En la página **User Groups**, localice la fila que contiene **user_group** y haga clic en **Authorize** en la columna **Operation** y seleccione **Policy1_IAM_OBS**, **Policy2_AllowOperation** y **Policy3_DenyOperation**. Haga clic en **Next**.
2. Especifique el ámbito como **All resources** y haga clic en **OK**.

Paso 4 Configure la autorización de acceso a ModelArts basada en agentes para que un desarrollador permita a ModelArts acceder a servicios dependientes como OBS.

1. Inicie sesión en la consola de ModelArts con una cuenta de tenant. En el panel de navegación de la izquierda, seleccione **Settings**. Se muestra la página **Global Configuration**.
2. Haga clic en **Add Authorization**. En la página **Add Authorization**, establezca **Authorized User** como **IAM user**, seleccione una cuenta de desarrollador para **Authorized To**, agregue una delegación **ma_agency_develop_user**, establezca **Permissions** como **Custom** y seleccione **OBS Administrator**. Los desarrolladores solo necesitan la autorización de OBS para permitirles acceder a OBS cuando utilizan notebook.
3. Haga clic en **Create**.
4. En la página **Global Configuration**, vuelva a hacer clic en **Add Authorization**. En la página **Add Authorization** que aparece en pantalla, configure una delegación para otros usuarios de desarrolladores.

   En la página **Add Authorization**, establezca **Authorized User** en **IAM user** y seleccione una cuenta de desarrollador para **Authorized To** y seleccione la delegación existente **ma_agency_develop_user** creada anteriormente.

Paso 5 Pruebe los permisos de desarrollador.

1. Inicie sesión en la consola de gestión de ModelArts como usuario de IAM en **user_group**. En la página de inicio de sesión, asegúrese de que **IAM User Login** esté seleccionado.

   Cambie la contraseña según se le indique al iniciar la sesión por primera vez.
2. En el panel de navegación de la consola de gestión de ModelArts, seleccione **Dedicated Resource Pools** y haga clic en **Create**. Se han asignado los permisos al desarrollador si la consola no muestra el mensaje de permisos insuficientes.

----Fin

2.3.4 Consulta de todas las instancias de notebook de un proyecto de IAM

Cualquier usuario de IAM que tenga los permisos **listAllNotebooks** y **listUsers** puede hacer clic en **View all** en la página de notebook para ver las instancias de todos los usuarios del proyecto de IAM actual.

> NOTA
>
> Los usuarios con estos permisos también pueden acceder a OBS y SWR de todos los usuarios del proyecto IAM actual.

Asignación de los permisos requeridos

1. Inicie sesión en la consola de gestión como usuario tenant, coloque el cursor sobre su nombre de usuario en la esquina superior derecha y elija **Identity and Access Management** en la lista desplegable para cambiar a la consola de gestión de IAM.
2. En la consola de IAM, elija **Permissions** > **Policies/Roles** en el panel de navegación, haga clic en **Create Custom Policy** en la esquina superior derecha y cree dos políticas.

   Política 1: Cree una política que permita a los usuarios ver todas las instancias de notebook de un proyecto de IAM.

- **Policy Name**: introduzca un nombre de política personalizado, por ejemplo, **Viewing all notebook instances**.
- **Policy View**: Seleccione **Visual editor**.
- **Policy Content**: Seleccione **Allow**, **ModelArts Service**, **modelarts:notebook:listAllNotebooks** y los recursos predeterminados.

Política 2: Cree una política que permita a los usuarios ver todos los usuarios de un proyecto IAM.

- **Policy Name**: introduzca un nombre de política personalizado, por ejemplo, **Viewing all users of the current IAM project**.
- **Policy View**: Seleccione **Visual editor**.
- **Policy Content**: Seleccione **Allow**, **Identity and Access Management**, **iam:users:listUsers** y los recursos predeterminados.

3. En el panel de navegación, seleccione **User Groups**. En la página **User Groups**, localice la fila que contiene el grupo de usuarios de destino y haga clic en **Authorize** en la columna **Operation**. En la página **Authorize User Group**, seleccione la política personalizada creada en **2** y haga clic en **Next**. A continuación, seleccione el ámbito y haga clic en **OK**.

   Después de la configuración, todos los usuarios del grupo de usuarios tienen permiso para ver todas las instancias de notebook creadas por los usuarios del grupo de usuarios.

   Si no hay ningún grupo de usuarios disponible, cree uno, agregue usuarios a él con la gestión de grupos de usuarios y configure la autorización para el grupo de usuarios. Si el usuario objetivo no pertenece a un grupo de usuarios, agréguelo a un grupo de usuarios con la gestión de grupos de usuarios.

Habilitación de un usuario de IAM para iniciar la instancia de notebook de otro usuario

Si un usuario de IAM desea acceder a la instancia de notebook de otro usuario de IAM a través de SSH remoto, debe actualizar el par de claves SSH a la suya. De lo contrario, se reportará el error **ModelArts.6786**. Para obtener más detalles sobre cómo actualizar un par de claves, consulte **Modificación de la configuración de SSH para una Instancia de notebook**.

ModelArts.6789: Error al encontrar el par de claves de SSH KeyPair-xxx en la página de pares de claves de ECS. Actualice el par de claves e inténtelo de nuevo más tarde.

2.3.5 Inicio de sesión en un contenedor de entrenamiento con Cloud Shell

Escenario de aplicación

Puede usar Cloud Shell proporcionado por la consola de ModelArts para iniciar sesión en un contenedor de entrenamiento en ejecución.

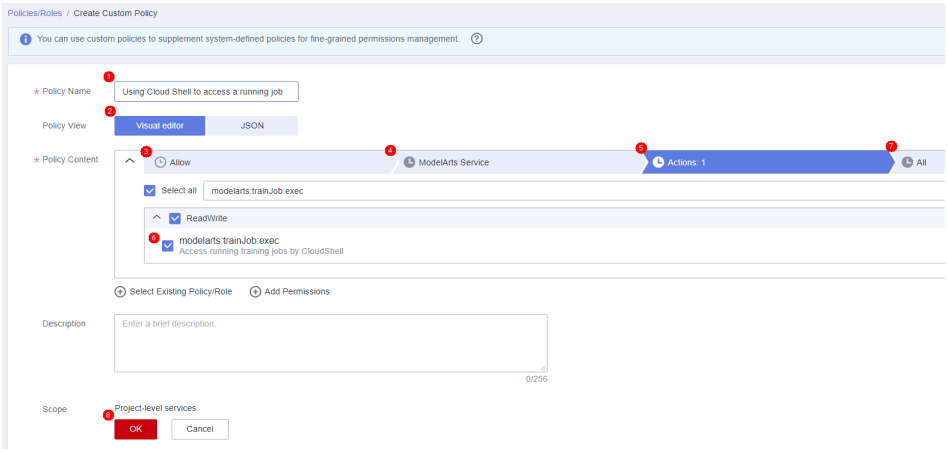
Restricciones

Solo los grupos de recursos dedicados admiten Cloud Shell. El trabajo de entrenamiento debe estar en el estado **Running**.

Preparación: Asignación del permiso de Cloud Shell a un usuario de IAM

1. Inicie sesión en la consola de gestión de Huawei Cloud como usuario tenant, coloque el cursor sobre su nombre de usuario en la esquina superior derecha y seleccione **Identity and Access Management** en la lista desplegable para pasar a la consola de gestión de IAM.
2. En la consola de IAM, elija **Permissions** > **Policies/Roles** en el panel de navegación, haga clic en **Create Custom Policy** en la esquina superior derecha y configure los siguientes parámetros.
   - **Policy Name**: introduzca un nombre de política personalizado, por ejemplo, **Using Cloud Shell to access a running job**.
   - **Policy View**: Seleccione **Visual editor**.
   - **Policy Content**: Seleccione **Allow**, **ModelArts Service**, **modelarts:trainJob:exec** y los recursos predeterminados.

   Figura 2-15 Creación de una política personalizada

3. En el panel de navegación, seleccione **User Groups**. A continuación, haga clic en **Authorize** en la columna **Operation** del grupo de usuarios de destino. En la página **Authorize User Group**, seleccione las políticas personalizadas creadas en **2** y haga clic en **Next**. A continuación, seleccione el ámbito y haga clic en **OK**.

   Después de la configuración, todos los usuarios del grupo de usuarios tienen el permiso para usar Cloud Shell para iniciar sesión en un contenedor de entrenamiento en ejecución.

   Si no hay ningún grupo de usuarios disponible, cree un grupo de usuarios, agregue usuarios mediante la función de gestión de grupos de usuarios y configure la autorización. Si el usuario de destino no está en un grupo de usuarios, puede agregarlo a un grupo de usuarios mediante la función de gestión de grupos de usuarios.

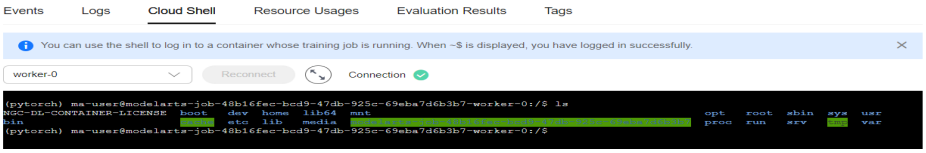
Uso de Cloud Shell

1. Configure los parámetros según **Preparación: Asignación del permiso de Cloud Shell a un usuario de IAM**.
2. En la consola de ModelArts, seleccione **Training Management** > **Training Jobs** en el panel de navegación.
3. En la lista de trabajos de entrenamiento, haga clic en el nombre del trabajo de destino para ir a la página de detalles del trabajo de entrenamiento.

4. En la página de detalles del trabajo de entrenamiento, haga clic en la ficha **Cloud Shell** e inicie sesión en el contenedor de entrenamiento.

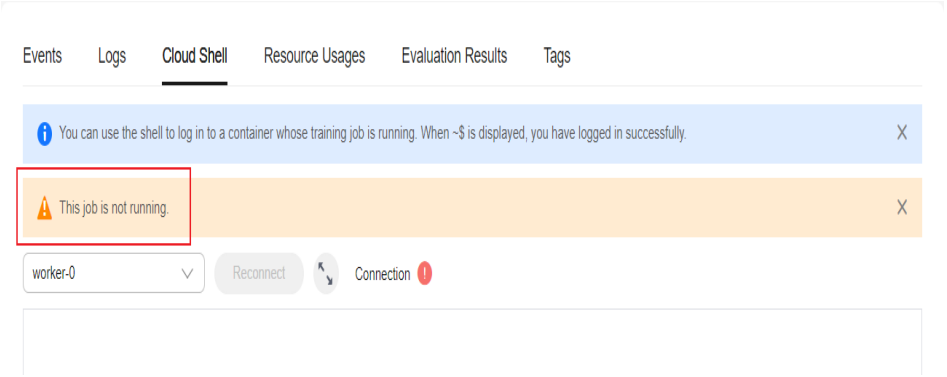
   Compruebe que el inicio de sesión se ha realizado correctamente, como se muestra en la siguiente figura.

Figura 2-16 Página de Cloud Shell



Si el trabajo no se está ejecutando o el permiso es insuficiente, no se puede utilizar Cloud Shell. En este caso, localice la falla como variable.

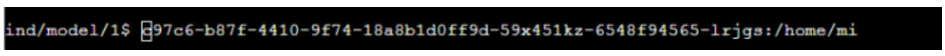
Figura 2-17 Mensaje de error



NOTA

Es posible que se produzca una excepción de visualización de rutas al iniciar sesión en la página de Cloud Shell. En este caso, presione **Enter** para detectar la falla.

Figura 2-18 Ruta anormal



2.3.6 Prohibición de que un usuario utilice un grupo de recursos público

Esta sección describe cómo controlar los permisos de ModelArts de un usuario para que no se le permita usar un grupo de recursos público para crear trabajos de entrenamiento, crear instancias de notebook o desplegar servicios de inferencia.

Contexto

Mediante el control de permisos, se puede prohibir a los usuarios del grupo de recursos dedicado de ModelArts que utilicen un grupo de recursos público para crear trabajos de entrenamiento, crear instancias de notebook o desplegar servicios de inferencia.

Para controlar los permisos, configure los siguientes elementos de políticas de permisos:

- **modelarts:notebook:create**: permite crear una instancia de notebook.
- **modelarts:trainJob:create**: permite crear un trabajo de entrenamiento.
- **modelarts:service:create**: permite crear un servicio de inferencia.

Procedimiento

1. Inicie sesión en la consola de gestión como usuario tenant, coloque el cursor sobre su nombre de usuario en la esquina superior derecha y elija **Identity and Access Management** en la lista desplegable para cambiar a la consola de gestión de IAM.
2. En el panel de navegación, seleccione **Permissions** > **Policies/Roles**. En la página **Policies/Roles**, haga clic en **Create Custom Policy** en la esquina superior derecha, configure los parámetros y haga clic en **OK**.
   - **Policy Name**: configure el nombre de la política.
   - **Policy View**: seleccione **Visual editor** o **JSON**.
   - **Policy Content**: seleccione **Deny**. En **Select service**, busque **ModelArts** y selecciónelo. En **ReadWrite** bajo **Actions**, busque **modelarts:trainJob:create**, **modelarts:notebook:create** y **modelarts:service:create** y selecciónelos. **All**: conserva la configuración predeterminada. En **Add request condition**, haga clic en **Add Request Condition**. En el cuadro de diálogo que aparece en pantalla, configure **Condition Key** a **modelarts:poolType**, **Operator** a **StringEquals** y **Value** to **public**.

     El contenido de la política en la vista de JSON es el siguiente:

     ```
     {
         "Version": "1.1",
         "Statement": [
             {
                 "Effect": "Deny",
                 "Action": [
                     "modelarts:trainJob:create",
                     "modelarts:notebook:create",
                     "modelarts:service:create"
                 ],
                 "Condition": {
                     "StringEquals": {
                         "modelarts:poolType": [
                             "public"
                         ]
                     }
                 }
             }
         ]
     }
     ```
3. En el panel de navegación, seleccione **User Groups**. En la página **User Groups**, localice la fila que contiene el grupo de usuarios de destino y haga clic en **Authorize** en la columna **Operation**. En la página **Authorize User Group**, seleccione la política personalizada creada en **2** y haga clic en **Next**. A continuación, seleccione el ámbito y haga clic en **OK**.

   Después de la configuración, todos los usuarios del grupo de usuarios tienen permiso para ver todas las instancias de notebook creadas por los usuarios del grupo de usuarios.

   Si no hay ningún grupo de usuarios disponible, cree uno, agregue usuarios a él con la gestión de grupos de usuarios y configure la autorización para el grupo de usuarios. Si el usuario objetivo no pertenece a un grupo de usuarios, agréguelo a un grupo de usuarios con la gestión de grupos de usuarios.
4. Agregue la política a la autorización de delegación del usuario. Esto evita que el usuario rompa el alcance del permiso con un token en el plano de tenant.

En el panel de navegación, elija **Agencies**. Localice la delegación utilizada por el grupo de usuarios de ModelArts y haga clic en **Modify** en la columna **Operation**. En la ficha **Permissions**, haga clic en **Authorize**, seleccione la política personalizada creada y haga clic en **Next**. Seleccione el ámbito de autorización y haga clic en **OK**.

Verificación

Inicie sesión en la consola de ModelArts como usuario de IAM, seleccione **Training Management** > **Training Jobs** y haga clic en **Create Training Job**. En la página para crear un trabajo de entrenamiento, solo se puede seleccionar un grupo de recursos dedicado para **Resource Pool**.

Inicie sesión en la consola de ModelArts como usuario de IAM, seleccione **DevEnviron** > **Notebook** y haga clic en **Create**. En la página para crear una instancia de notebook, solo se puede seleccionar un grupo de recursos dedicado para **Resource Pool**.

Inicie sesión en la consola de ModelArts como usuario de IAM, seleccione **Service Deployment** > **Real-Time Services** y haga clic en **Deploy**. En la página de despliegue de servicio, solo se puede seleccionar un grupo de recursos dedicado para **Resource Pool**.

2.3.7 Concesión de permisos de acceso a la carpeta SFS Turbo a usuarios de IAM

Escenarios

Otorgue permisos de acceso a las carpetas de SFS Turbo específicas a usuarios de IAM.

Restricciones

- Asegúrese de haber habilitado una autorización estricta. Inicie sesión en la consola de ModelArts y seleccione **Settings** en el panel de navegación de la izquierda. En la página **Global Configuration**, haga clic en **Enable strict authorization**.
- Si no se ha otorgado los permisos de ModelArts a los usuarios de IAM, es posible que los usuarios no puedan utilizar ModelArts una vez activada la autorización estricta. Otorgue el permiso a los usuarios de IAM consultando **Asignación de permisos a usuarios individuales para usar ModelArts**.

Procedimiento

Paso 1 Inicie sesión en la consola de gestión con la cuenta principal, coloque el cursor sobre su nombre de usuario en la esquina superior derecha y seleccione **Identity and Access Management** en la lista desplegable para pasar a la consola de gestión de IAM.

Paso 2 En la consola de IAM, seleccione **Permissions** > **Policies/Roles** en el panel de navegación de la izquierda, haga clic en **Create Custom Policy** en la esquina superior derecha y configure la política de la siguiente manera:

- **Policy Name**: introduzca un nombre de política, por ejemplo, **ma_sfs_turbo**.
- **Policy View**: seleccione **JSON**.
- **Policy Content**: Introduzca la siguiente información:

```
{
    "Version": "1.1",
    "Statement": [
        {
```



```
            "Effect": "Allow",
            "Action": [
                "<modelarts_action>"
            ],
            "Condition": {
                "StringEquals": {
                    "modelarts:sfsId": [
                        "<your_ssf_id>"
                    ],
                    "modelarts:sfsPath": [
                        "<sfs_path>"
                    ],
                    "modelarts:sfsOption": [
                        "<sfs_option>"
                    ]
                }
            }
        }
    ]
}
```

Sustituya <*modelarts_action*>, <*your_ssf_id*>, <*sfs_path*> y <*sfs_option*> con los parámetros reales que necesite. La tabla siguiente describe los parámetros.

Tabla 2-22 Descripción del parámetro

Parámetro	Descripción
Action	Escenario en el que se otorga el permiso de acceso a la carpeta de SFS Turbo. ● **modelarts:trainJob:create**: El permiso se concede durante la creación de la instancia del entorno de desarrollo. ● **modelarts:notebook:create**: el permiso se concede durante la creación del trabajo de entrenamiento. Se soportan múltiples acciones. A continuación se muestra un ejemplo: `"Action": [ "modelarts:trainJob:create", "modelarts:notebook:create" ],`
modelarts:sfsId	ID de SFS Turbo, que puede obtenerse en la página de detalles de SFS Turbo. Puede introducir varios ID. A continuación se muestra un ejemplo: `"modelarts:sfsId": [ "0e51c7d5-d90e-475a-b5d0-ecf896da3b0d", "2a70da1e-ea87-4ee4-ae1e-55df846e7f41" ],`

Parámetro	Descripción

modelarts:sfsPath	Ruta de la carpeta de SFS Turbo cuyos permisos deben configurarse. Puede introducir varias rutas. A continuación se muestra un ejemplo:
	`"modelarts:sfsPath": [`
	`  "/path1",`
	`  "/path2/path2-1"`
	`],`
	Si existen varios ID de SFS, las rutas de SFS se aplicarán a todos los ID de SFS. Como se muestra en el siguiente ejemplo, se configura el permiso para acceder a **/path1** y **/path2/path2-1** de ambos **0e51c7d5-d90e-475a-b5d0-ecf896da3b0d** y **2a70da1e-ea87-4ee4-ae1e-55df846e7f41**.
	`"modelarts:sfsId": [`
	`  "0e51c7d5-d90e-475a-b5d0-ecf896da3b0d",`
	`  "2a70da1e-ea87-4ee4-ae1e-55df846e7f41"`
	`],`
	`"modelarts:sfsPath": [`
	`  "/path1",`
	`  "/path2/path2-1"`
	`],`

Parámetro	Descripción
modelarts:sfsOption	Tipo de permiso de acceso. Se soportan los siguientes parámetros: ● **readonly**: permiso de solo lectura ● **readwrite**: permiso de lectura y escritura Para agregar varias opciones de SFS a una política personalizada, agregue una estructura de JSON a Statement, a continuación se muestra un ejemplo: `{ "Version": "1.1", "Statement": [{ "Effect": "Allow", "Action": ["modelarts:notebook:create"], "Condition": { "StringEquals": { "modelarts:sfsId": ["0e51c7d5-d90e-475a-b5d0-ecf896da3b0d"], "modelarts:sfsPath": ["/path1"], "modelarts:sfsOption": ["readonly"] } } }, { "Effect": "Allow", "Action": ["modelarts:notebook:create"], "Condition": { "StringEquals": { "modelarts:sfsId": ["0e51c7d5-d90e-475a-b5d0-ecf896da3b0d"], "modelarts:sfsPath": ["/path2"], "modelarts:sfsOption": ["readwrite"] } } }] }`

Paso 3 Cree un grupo de usuarios y agregue el usuario a este grupo. Consulte **Paso 1 Cree un grupo de usuarios y agregue datos al grupo de usuarios** para obtener más detalles.

Paso 4 Otorgue una política al grupo de usuarios. En la página de lista de grupos de usuarios de IAM, haga clic en **Authorize** del grupo de usuarios de destino. Aparecerá la página **Authorize User Group**. Seleccione la política **ma_sfs_turbo** creada en el **Paso 2**. Haga clic en **Next** y luego en **OK**.

Paso 5 Agregue el permiso **IAM ReadOnlyAccess** a una delegación de ModelArts existente.

1. En la consola de gestión de ModelArts, seleccione **Settings** en el panel de navegación de la izquierda. En la página que aparece en pantalla, busque la delegación de destino, seleccione **View Permissions** en la columna **Operation** y haga clic en **Modify permission in IAM**.
2. En la consola de IAM, seleccione **Agencies** en el panel de navegación situado a la izquierda y luego seleccione **Permissions** > **Authorize**. Busque **IAM ReadOnlyAccess**, actívelo y haga clic en **Next** y **OK**.

Paso 6 Verifique que se haya concedido el permiso.

Inicie sesión en ModelArts como usuario IAM, solo se mostrarán las carpetas de SFS Turbo configuradas durante la creación de trabajos de entrenamiento y la creación de notebook.

----Fin

2.4 Preguntas frecuentes

2.4.1 ¿Qué debo hacer si se muestra un mensaje que indica permisos insuficientes cuando utilizo ModelArts?

Si aparece un mensaje que indica que no hay permisos suficientes cuando utiliza ModelArts, realice las operaciones descritas en esta sección para conceder permisos para los servicios relacionados según sea necesario.

Los permisos para usar ModelArts dependen de la autorización de OBS. Por lo tanto, los usuarios de ModelArts también requieren permisos del sistema de OBS.

- Para obtener más información acerca de cómo conceder a un usuario permisos completos para OBS y permisos de operaciones comunes para ModelArts, consulte **Configuración de permisos de operaciones comunes**.
- Para obtener más información acerca de cómo gestionar los permisos de usuario en OBS y ModelArts de forma precisa y configurar las políticas personalizadas, consulte **Creación de una política personalizada para ModelArts**.

Configuración de permisos de operaciones comunes

Para utilizar las funciones básicas de ModelArts, asigne a los usuarios el permiso **ModelArts CommonOperations** en los servicios de nivel de proyecto. Dado que ModelArts depende de los permisos de OBS, asigne a los usuarios el permiso de **OBS Administrator** en los servicios globales.

El procedimiento es el siguiente:

Paso 1 Cree un grupo de usuarios.

Inicie sesión en la consola de IAM y elija **User Groups** > **Create User Group**. Escriba un nombre de grupo de usuarios y haga clic en **OK**.

Paso 2 Configure permisos para el grupo de usuarios.

En la lista de grupos de usuarios, localice el grupo de usuarios creado en el paso 1, haga clic en **Authorize** y realice las siguientes operaciones.

1. Asigne el permiso **ModelArts CommonOperations** sobre los servicios a nivel de proyecto para el grupo de usuarios y haga clic en **OK**.

> **NOTA**
>
> El permiso solo tiene efecto en las regiones asignadas. Asigne permisos en todas las regiones si en todas es necesario.

2. Asigne el permiso **OBS Administrator** en los servicios globales para el grupo de usuarios y haga clic en **OK**.

Paso 3 **Cree un usuario y agréguelo a un grupo de usuarios.**

Cree un usuario en la consola de IAM y agréguelo al grupo de usuarios creado en el paso 1.

Paso 4 **Inicie sesión como usuario de IAM** y verifique los permisos.

Inicie sesión en la consola de ModelArts como el usuario creado, cambie a la región autorizada y compruebe que las políticas **ModelArts CommonOperations** y **Tenant Administrator** están en vigor.

- Elija **Service List** > **ModelArts**. Elija **Dedicated Resource Pools**. En la página que se muestra, seleccione un tipo de grupo de recursos y haga clic en **Create**. No debería poder crear un nuevo grupo de recursos.
- Elija cualquier otro servicio en **Service List**. (Supongamos que la política actual solo contiene **ModelArts CommonOperations**) Si aparece un mensaje indicando que no tiene permisos suficientes para acceder al servicio, la política **ModelArts CommonOperations** ya ha surtido efecto.
- Elija **Service List** > **ModelArts**. En la consola de ModelArts, elija **Data Management** > **Datasets** > **Create Dataset**. Debería poder acceder a la ruta de OBS correspondiente.

----Fin

Creación de una política personalizada para ModelArts

Además de las políticas de sistema predeterminadas de ModelArts, puede crear políticas personalizadas, que también pueden abordar los permisos de OBS. Para obtener más información, consulte **Creación de una política personalizada**.

Puede crear políticas personalizadas mediante las vistas de visual editor o JSON. Esta sección describe cómo utilizar un archivo de JSON a conceder a un usuario permisos para acceder a un entorno de desarrollo y a conceder a ModelArts los permisos mínimos necesarios para acceder a OBS.

> **NOTA**
>
> Una política personalizada puede contener acciones para varios servicios a los que se puede acceder globalmente o solo para los proyectos específicos de la región.
>
> ModelArts es un servicio a nivel de proyecto, pero OBS es un servicio global, por lo que debe crear políticas separadas para los dos servicios y luego aplicar estas políticas a los usuarios.

1. Crear una política personalizada para minimizar los permisos para OBS del que depende ModelArts.

   Inicie sesión en la consola de IAM, seleccione **Permissions** > **Policies/Roles** y haga clic en **Create Custom Policy**. Configure los parámetros de la siguiente manera:

   – **Policy Name**: Seleccione un nombre de la política personalizada.
   – **Policy View**: **JSON**

- **Policy Content**: Siga las instrucciones de **Ejemplo de políticas personalizadas de OBS**. Para obtener más información sobre los permisos del sistema de OBS, consulte **Gestión de permisos de OBS**.

2. Cree una política personalizada para los permisos de usar los entornos de desarrollo de ModelArts. Configure los parámetros de la siguiente manera:
   - **Policy Name**: Seleccione un nombre de la política personalizada.
   - **Policy View**: **JSON**
   - **Policy Content**: Siga las instrucciones de **Ejemplo de políticas personalizadas para utilizar el entorno de desarrollo de ModelArts**. Para conocer las acciones que pueden agregarse a las directivas personalizadas, consulte **Referencia de las API de ModelArts > Políticas de permisos y acciones admitidas**.
   - Para ver las políticas del sistema de otros servicios, consulte **Permisos del sistema**.
3. **Cree un grupo de usuarios y conceder permisos al grupo de usuarios** en la consola de IAM.

   Después de crear un grupo de usuarios en la consola de IAM, conceda la política personalizada creada en **1** al grupo de usuarios.
4. **Cree un usuario y agréguelo a un grupo de usuarios.**

   Cree un usuario en la consola de IAM y agregue el usuario al grupo creado en **3**.
5. **Inicie sesión como usuario de IAM** y verifique los permisos.

   Inicie sesión en la consola de ModelArts como el usuario creado, cambie a la región autorizada y compruebe que las políticas **ModelArts CommonOperations** y **Tenant Administrator** están en vigor.
   - Elija **Service List** > **ModelArts**. En la consola de ModelArts, elija **Data Management** > **Datasets**. Si no puede crear un conjunto de datos, los permisos (para usar el entorno de desarrollo) concedidos solo a los usuarios de ModelArts han surtido efecto.
   - Elija **Service List** > **ModelArts**. En la consola de ModelArts, seleccione **DevEnviron** > **Notebook** y haga clic en **Create**. Si puede acceder a la ruta del OBS especificada en **Storage**, los permisos de OBS ha entrado en vigor.

Ejemplo de políticas personalizadas de OBS

Los permisos para usar ModelArts requieren la autorización de OBS. En el ejemplo siguiente se muestra el OBS mínimo necesario, incluidos los permisos para los bucket y objetos de OBS. Después de obtener los permisos mínimos para OBS, los usuarios pueden acceder a OBS desde ModelArts sin restricciones.

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Action": [
                "obs:bucket:ListAllMybuckets",
                "obs:bucket:HeadBucket",
                "obs:bucket:ListBucket",
                "obs:bucket:GetBucketLocation",
                "obs:object:GetObject",
                "obs:object:GetObjectVersion",
                "obs:object:PutObject",
                "obs:object:DeleteObject",
                "obs:object:DeleteObjectVersion",
                "obs:object:ListMultipartUploadParts",
                "obs:object:AbortMultipartUpload",
```



```
                "obs:object:GetObjectAcl",
                "obs:object:GetObjectVersionAcl",
                "obs:bucket:PutBucketAcl",
                "obs:object:PutObjectAcl"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Ejemplo de políticas personalizadas para utilizar el entorno de desarrollo de ModelArts

```
{
    "Version": "1.1",
    "Statement": [

        {
            "Effect": "Allow",
            "Action": [
                "modelarts:notebook:list",
                "modelarts:notebook:create" ,
                "modelarts:notebook:get" ,
                "modelarts:notebook:update" ,
                "modelarts:notebook:delete" ,
                "modelarts:notebook:action" ,
                "modelarts:notebook:access"
            ]
        }
    ]
}
```


3 Notebook

3.1 Creación, migración y gestión de entornos virtuales de Conda basados en SFS

En este tema se describe cómo migrar el entorno de Conda de una instancia de notebook a un disco de SFS. De esta manera, el entorno de Conda no se perderá después de reiniciar la instancia de notebook.

El procedimiento es el siguiente:

1. **Crear un entorno virtual y guardarlo en el directorio de SFS**
2. **Clonación de los entornos virtuales existentes en el disco de SFS**
3. **Reinicio de la imagen para activar el entorno virtual en el disco de SFS**
4. **Guardar y compartir el entorno virtual**

Requisitos previos

Ha creado una instancia de notebook configurando **Resource Type** en **Dedicated resource pool** y **Storage** en **SFS** y abierto el terminal.

Crear un entorno virtual y guardarlo en el directorio de SFS

Crear un entorno virtual de conda.

```
# shell
conda create --prefix /home/ma-user/work/envs/user_conda/sfs-new-env
python=3.7.10 -y
```

Consulte los entornos virtuales de conda existentes. El nombre del entorno virtual recién creado puede estar vacío en la salida.

```
# shell
conda env list
# conda environments:
#
base                     /home/ma-user/anaconda3
PyTorch-1.8              /home/ma-user/anaconda3/envs/PyTorch-1.8
python-3.7.10         *  /home/ma-user/anaconda3/envs/python-3.7.10
                         /home/ma-user/work/envs/user_conda/sfs-new-env
```


Agregue el nuevo entorno virtual a conda envs.

```
# shell
conda config --append envs_dirs /home/ma-user/work/envs/user_conda/
```

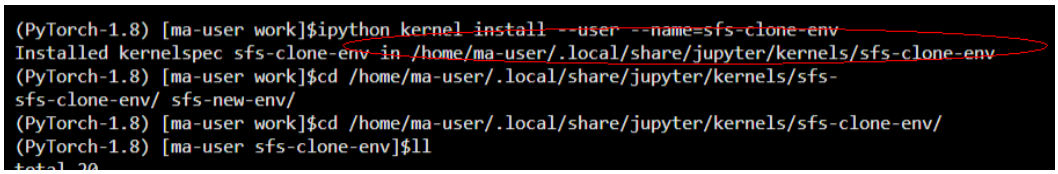
Consulte los entornos virtuales de conda existentes. El nuevo entorno virtual se muestra correctamente y puede cambiar a él por su nombre.

```
# shell
conda env list
conda activate sfs-new-env
# conda environments:
#
base                     /home/ma-user/anaconda3
PyTorch-1.8              /home/ma-user/anaconda3/envs/PyTorch-1.8
python-3.7.10         *  /home/ma-user/anaconda3/envs/python-3.7.10
sfs-new-env              /home/ma-user/work/envs/user_conda/sfs-new-env
```

(Opcional) Registre el nuevo entorno virtual con JupyterLab kernel para poder utilizarlo directamente en JupyterLab.

```
# shell
pip install ipykernel
ipython kernel install --user --name=sfs-new-env
rm -rf /home/ma-user/.local/share/jupyter/kernels/sfs-new-env/logo-*
```

Nota: **.local/share/jupyter/kernels/sfs-new-env** se utiliza solo como ejemplo. Hágalo con la ruta de instalación real.



Actualice la página de JupyterLab. Se muestra el nuevo kernel.

NOTA

Una vez reiniciada la instancia de notebook, es necesario volver a registrar el kernel.

Clonación de los entornos virtuales existentes en el disco de SFS

```
# shell
conda create --prefix /home/ma-user/work/envs/user_conda/sfs-clone-env --
clone PyTorch-1.8 -y
Source:      /home/ma-user/anaconda3/envs/PyTorch-1.8
Destination: /home/ma-user/work/envs/user_conda/sfs-clone-env
Packages: 20
Files: 39687
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate /home/ma-user/work/envs/user_conda/sfs-clone-env
#
# To deactivate an active environment, use
#
#     $ conda deactivate
```

Consulte los entornos virtuales clonados. Si el nombre del entorno virtual recién creado está vacío, maneje el problema según **Agregue el nuevo entorno virtual al conda envs**.


```
# shell
conda env list
# conda environments:
#
base                     /home/ma-user/anaconda3
PyTorch-1.8              /home/ma-user/anaconda3/envs/PyTorch-1.8
python-3.7.10            /home/ma-user/anaconda3/envs/python-3.7.10
sfs-clone-env            /home/ma-user/work/envs/user_conda/sfs-clone-env
sfs-new-env           *  /home/ma-user/work/envs/user_conda/sfs-new-env
```

(Opcional) Registre el nuevo entorno virtual con JupyterLab kernel para poder utilizarlo directamente en JupyterLab.

```
# shell
pip install ipykernel
ipython kernel install --user --name=sfs-clone-env
rm -rf /home/ma-user/.local/share/jupyter/kernels/sfs-clone-env/logo-*
```

Note: **.local/share/jupyter/kernels/sfs-clone-env** is used as an example only. Replace it with the actual installation path.

Actualice la página de JupyterLab. Se muestra el nuevo kernel.

Reinicio de la imagen para activar el entorno virtual en el disco de SFS

Método 1: Utilice la ruta completa de conda env.

```
# shell
conda activate /home/ma-user/work/envs/user_conda/sfs-new-env
```

Método 2: Agregue el entorno virtual a conda envs y actívelo con su nombre.

```
# shell
conda config --append envs_dirs /home/ma-user/work/envs/user_conda/
conda activate sfs-new-env
```

Método 3: Utilice Python o pip en el entorno virtual.

```
# shell
/home/ma-user/work/envs/user_conda/sfs-new-env/bin/pip list
/home/ma-user/work/envs/user_conda/sfs-new-env/bin/python -V
```

Guardar y compartir el entorno virtual

Empaquetar el entorno virtual que se va a migrar.

```
# shell
pip install conda-pack
conda pack -n sfs-clone-env -o sfs-clone-env.tar.gz --ignore-editable-
packages
Collecting packages...
Packing environment at '/home/ma-user/work/envs/user_conda/sfs-clone-env' to 'sfs-
clone-env.tar.gz'
[########################################] | 100% Completed |  3min 33.9s
```

Descomprima el paquete en el directorio de SFS.

```
# shell

mkdir /home/ma-user/work/envs/user_conda/sfs-tar-env
tar -zxvf sfs-clone-env.tar.gz -C /home/ma-user/work/envs/user_conda/sfs-
tar-env
```


Consulte los entornos virtuales de conda existentes.

```
# shell
conda env list
# conda environments:
#
base                     /home/ma-user/anaconda3
PyTorch-1.8           *  /home/ma-user/anaconda3/envs/PyTorch-1.8
python-3.7.10            /home/ma-user/anaconda3/envs/python-3.7.10
sfs-clone-env            /home/ma-user/work/envs/user_conda/sfs-clone-env
sfs-new-env              /home/ma-user/work/envs/user_conda/sfs-new-env
sfs-tar-env              /home/ma-user/work/envs/user_conda/sfs-tar-env
test-env                 /home/ma-user/work/envs/user_conda/test-env
```


4 Entrenamiento de modelos

4.1 Uso de un algoritmo personalizado para crear un modelo de reconocimiento de dígitos escrito a mano

En esta sección se describe cómo modificar un algoritmo local personalizado para entrenar y desplegar los modelos de ModelArts.

Escenarios

En este caso se describe cómo utilizar PyTorch 1.8 para reconocer imágenes de dígitos manuales. En este caso, se utiliza un conjunto de datos oficial del MNIST.

A través de este caso, puede aprender a entrenar trabajos, desplegar un modelo de inferencia y realizar predicciones sobre ModelArts.

Proceso

Antes de realizar las siguientes operaciones, consulte **Preparaciones** para completar las operaciones necesarias.

1. **Paso 1 Preparar los datos de entrenamiento**: Descargar el conjunto de datos del MNIST.
2. **Paso 2 Preparar los archivos de entrenamiento y de inferencia**: Escribir código de entrenamiento y de inferencia.
3. **Paso 3 Crear un bucket de OBS y cargar archivos a OBS**: Crear un bucket y una carpeta de OBS y cargue el conjunto de datos, el script de entrenamiento, el script de inferencia y el archivo de configuración de inferencia en OBS.
4. **Paso 4 Crear un trabajo de entrenamiento**: Entrenar un modelo.
5. **Paso 5: Desplegar el modelo para la inferencia**: importar el modelo entrenado a ModelArts, crear una aplicación de IA y desplegar la aplicación de IA como un servicio en tiempo real.
6. **Paso 6 Realizar una predicción**: cargar una imagen de dígitos manuales y enviar una solicitud de inferencia para obtener el resultado de la inferencia.
7. **Paso 7 Lanzar recursos**: Detener el servicio y eliminar los datos del OBS para detener la facturación.

Preparaciones

- Ha creado un ID de Huawei y ha habilitado los servicios en Huawei Cloud. Además, la cuenta no está en mora ni congelada.
- Configurar una delegación.

  Para utilizar ModelArts, se requiere acceso al Object Storage Service (OBS), SoftWare Repository for Container (SWR) e Intelligent EdgeFabric (IEF). Si es la primera vez que utiliza ModelArts, configure una delegación para autorizar el acceso a estos servicios.

  a. Inicie sesión en la **consola de ModelArts** con su cuenta de Huawei Cloud. En el panel de navegación de la izquierda, seleccione **Settings**. En la página **Global Configuration**, haga clic en **Add Authorization**.

  b. Configure los parámetros de la siguiente manera en la página mostrada:

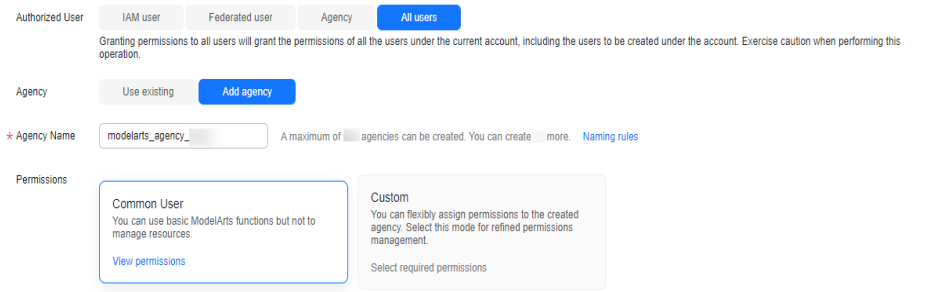
  Authorized User: **All users**.

  Agency: **Add agency**.

  Permissions: **Common User**.

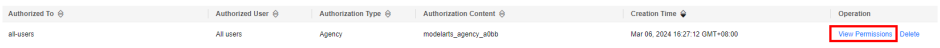
  Seleccione "I have read and agree to the ModelArts Service Statement" (He leído y acepto la declaración de servicio de ModelArts) y haga clic en **Create**.

  Figura 4-1 Configuración de una delegación



  c. Después de la configuración, consulte las configuraciones de delegación de su cuenta en la página **Global Configuration**.

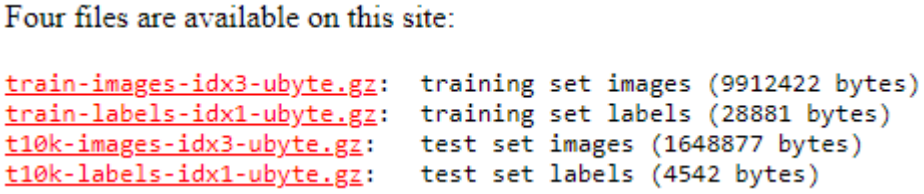
  Figura 4-2 Consulta de configuraciones de agencias



Paso 1 Preparar los datos de entrenamiento

En este caso se utiliza un conjunto de datos del MNIST descargado del **sitio web oficial de MNIST**. Asegúrese de que los cuatro archivos estén todos descargados.

Figura 4-3 Conjunto de datos del MNIST



- **train-images-idx3-ubyte.gz**: paquete comprimido del conjunto de entrenamiento, que contiene 60,000 muestras.
- **train-labels-idx1-ubyte.gz**: paquete comprimido de las etiquetas del conjunto de entrenamiento, que contiene las etiquetas de 60,000 muestras
- **t10k-images-idx3-ubyte.gz**: paquete comprimido del conjunto de validación, que contiene 10,000 muestras.
- **t10k-labels-idx1-ubyte.gz**: paquete comprimido de las etiquetas del conjunto de validación, que contiene las etiquetas de 10,000 muestras

NOTA

Si se le pide que introduzca la información de inicio de sesión después de hacer clic en el enlace del sitio web oficial del MNIST, copie y pegue este enlace en el cuadro de direcciones de su navegador: http://yann.lecun.com/exdb/mnist/.

La información de inicio de sesión es necesaria cuando se abre el enlace en modo de HTTPS, que no es necesaria si se abre el enlace en modo de HTTP.

Paso 2 Preparar los archivos de entrenamiento y de inferencia

En este caso, ModelArts proporciona el script de entrenamiento, el script de inferencia y el archivo de configuración de inferencia.

NOTA

Cuando pegue código desde un archivo .py, cree un archivo .py. De lo contrario, se puede aparece el mensaje de error "SyntaxError: 'gbk' codec can't decode byte 0xa4 in position 324: illegal multibyte sequence".

Cree el script de entrenamiento **train.py** en el host local. El contenido es el siguiente:

```
# base on https://github.com/pytorch/examples/blob/main/mnist/main.py

from __future__ import print_function

import os
import gzip
import codecs
import argparse
from typing import IO, Union

import numpy as np

import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torchvision import datasets, transforms
from torch.optim.lr_scheduler import StepLR

import shutil


# Define a network model.
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, 3, 1)
        self.conv2 = nn.Conv2d(32, 64, 3, 1)
        self.dropout1 = nn.Dropout(0.25)
        self.dropout2 = nn.Dropout(0.5)
        self.fc1 = nn.Linear(9216, 128)
        self.fc2 = nn.Linear(128, 10)
```



```
    def forward(self, x):
        x = self.conv1(x)
        x = F.relu(x)
        x = self.conv2(x)
        x = F.relu(x)
        x = F.max_pool2d(x, 2)
        x = self.dropout1(x)
        x = torch.flatten(x, 1)
        x = self.fc1(x)
        x = F.relu(x)
        x = self.dropout2(x)
        x = self.fc2(x)
        output = F.log_softmax(x, dim=1)
        return output


# Train the model. Set the model to the training mode, load the training data,
calculate the loss function, and perform gradient descent.
def train(args, model, device, train_loader, optimizer, epoch):
    model.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        data, target = data.to(device), target.to(device)
        optimizer.zero_grad()
        output = model(data)
        loss = F.nll_loss(output, target)
        loss.backward()
        optimizer.step()
        if batch_idx % args.log_interval == 0:
            print('Train Epoch: {} [{}/{} ({:.0f}%)]\tLoss: {:.6f}'.format(
                epoch, batch_idx * len(data), len(train_loader.dataset),
                100. * batch_idx / len(train_loader), loss.item()))
            if args.dry_run:
                break


# Validate the model. Set the model to the validation mode, load the validation
data, and calculate the loss function and accuracy.
def test(model, device, test_loader):
    model.eval()
    test_loss = 0
    correct = 0
    with torch.no_grad():
        for data, target in test_loader:
            data, target = data.to(device), target.to(device)
            output = model(data)
            test_loss += F.nll_loss(output, target, reduction='sum').item()
            pred = output.argmax(dim=1, keepdim=True)
            correct += pred.eq(target.view_as(pred)).sum().item()

    test_loss /= len(test_loader.dataset)

    print('\nTest set: Average loss: {:.4f}, Accuracy: {}/{} ({:.0f}%)\n'.format(
        test_loss, correct, len(test_loader.dataset),
        100. * correct / len(test_loader.dataset)))


# The following is PyTorch MNIST.
# https://github.com/pytorch/vision/blob/v0.9.0/torchvision/datasets/mnist.py
def get_int(b: bytes) -> int:
    return int(codecs.encode(b, 'hex'), 16)


def open_maybe_compressed_file(path: Union[str, IO]) -> Union[IO, gzip.GzipFile]:
    """Return a file object that possibly decompresses 'path' on the fly.
       Decompression occurs when argument `path` is a string and ends with '.gz'
or '.xz'.
    """
    if not isinstance(path, torch._six.string_classes):
```



```
        return path
    if path.endswith('.gz'):
        return gzip.open(path, 'rb')
    if path.endswith('.xz'):
        return lzma.open(path, 'rb')
    return open(path, 'rb')


SN3_PASCALVINCENT_TYPEMAP = {
    8: (torch.uint8, np.uint8, np.uint8),
    9: (torch.int8, np.int8, np.int8),
    11: (torch.int16, np.dtype('>i2'), 'i2'),
    12: (torch.int32, np.dtype('>i4'), 'i4'),
    13: (torch.float32, np.dtype('>f4'), 'f4'),
    14: (torch.float64, np.dtype('>f8'), 'f8')
}


def read_sn3_pascalvincent_tensor(path: Union[str, IO], strict: bool = True) ->
torch.Tensor:
    """Read a SN3 file in "Pascal Vincent" format (Lush file 'libidx/idx-io.lsh').
       Argument may be a filename, compressed filename, or file object.
    """
    # read
    with open_maybe_compressed_file(path) as f:
        data = f.read()
    # parse
    magic = get_int(data[0:4])
    nd = magic % 256
    ty = magic // 256
    assert 1 <= nd <= 3
    assert 8 <= ty <= 14
    m = SN3_PASCALVINCENT_TYPEMAP[ty]
    s = [get_int(data[4 * (i + 1): 4 * (i + 2)]) for i in range(nd)]
    parsed = np.frombuffer(data, dtype=m[1], offset=(4 * (nd + 1)))
    assert parsed.shape[0] == np.prod(s) or not strict
    return torch.from_numpy(parsed.astype(m[2], copy=False)).view(*s)


def read_label_file(path: str) -> torch.Tensor:
    with open(path, 'rb') as f:
        x = read_sn3_pascalvincent_tensor(f, strict=False)
    assert(x.dtype == torch.uint8)
    assert(x.ndimension() == 1)
    return x.long()


def read_image_file(path: str) -> torch.Tensor:
    with open(path, 'rb') as f:
        x = read_sn3_pascalvincent_tensor(f, strict=False)
    assert(x.dtype == torch.uint8)
    assert(x.ndimension() == 3)
    return x


def extract_archive(from_path, to_path):
    to_path = os.path.join(to_path, os.path.splitext(os.path.basename(from_path))
[0])
    with open(to_path, "wb") as out_f, gzip.GzipFile(from_path) as zip_f:
        out_f.write(zip_f.read())
# The above is pytorch mnist.
# --- end


# Raw MNIST dataset processing
def convert_raw_mnist_dataset_to_pytorch_mnist_dataset(data_url):
    """
    raw
```



```
    {data_url}/
        train-images-idx3-ubyte.gz
        train-labels-idx1-ubyte.gz
        t10k-images-idx3-ubyte.gz
        t10k-labels-idx1-ubyte.gz

    processed

    {data_url}/
        train-images-idx3-ubyte.gz
        train-labels-idx1-ubyte.gz
        t10k-images-idx3-ubyte.gz
        t10k-labels-idx1-ubyte.gz
        MNIST/raw
            train-images-idx3-ubyte
            train-labels-idx1-ubyte
            t10k-images-idx3-ubyte
            t10k-labels-idx1-ubyte
        MNIST/processed
            training.pt
            test.pt
    """
    resources = [
        "train-images-idx3-ubyte.gz",
        "train-labels-idx1-ubyte.gz",
        "t10k-images-idx3-ubyte.gz",
        "t10k-labels-idx1-ubyte.gz"
    ]

    pytorch_mnist_dataset = os.path.join(data_url, 'MNIST')

    raw_folder = os.path.join(pytorch_mnist_dataset, 'raw')
    processed_folder = os.path.join(pytorch_mnist_dataset, 'processed')

    os.makedirs(raw_folder, exist_ok=True)
    os.makedirs(processed_folder, exist_ok=True)

    print('Processing...')

    for f in resources:
        extract_archive(os.path.join(data_url, f), raw_folder)

    training_set = (
        read_image_file(os.path.join(raw_folder, 'train-images-idx3-ubyte')),
        read_label_file(os.path.join(raw_folder, 'train-labels-idx1-ubyte'))
    )
    test_set = (
        read_image_file(os.path.join(raw_folder, 't10k-images-idx3-ubyte')),
        read_label_file(os.path.join(raw_folder, 't10k-labels-idx1-ubyte'))
    )
    with open(os.path.join(processed_folder, 'training.pt'), 'wb') as f:
        torch.save(training_set, f)
    with open(os.path.join(processed_folder, 'test.pt'), 'wb') as f:
        torch.save(test_set, f)

    print('Done!')


def main():
    # Define the preset running parameters of the training job.
    parser = argparse.ArgumentParser(description='PyTorch MNIST Example')

    parser.add_argument('--data_url', type=str, default=False,
                        help='mnist dataset path')
    parser.add_argument('--train_url', type=str, default=False,
                        help='mnist model path')

    parser.add_argument('--batch-size', type=int, default=64, metavar='N',
                        help='input batch size for training (default: 64)')
```



```
    parser.add_argument('--test-batch-size', type=int, default=1000, metavar='N',
                        help='input batch size for testing (default: 1000)')
    parser.add_argument('--epochs', type=int, default=14, metavar='N',
                        help='number of epochs to train (default: 14)')
    parser.add_argument('--lr', type=float, default=1.0, metavar='LR',
                        help='learning rate (default: 1.0)')
    parser.add_argument('--gamma', type=float, default=0.7, metavar='M',
                        help='Learning rate step gamma (default: 0.7)')
    parser.add_argument('--no-cuda', action='store_true', default=False,
                        help='disables CUDA training')
    parser.add_argument('--dry-run', action='store_true', default=False,
                        help='quickly check a single pass')
    parser.add_argument('--seed', type=int, default=1, metavar='S',
                        help='random seed (default: 1)')
    parser.add_argument('--log-interval', type=int, default=10, metavar='N',
                        help='how many batches to wait before logging training
status')
    parser.add_argument('--save-model', action='store_true', default=True,
                        help='For Saving the current Model')
    args = parser.parse_args()

    use_cuda = not args.no_cuda and torch.cuda.is_available()

    torch.manual_seed(args.seed)

    # Set whether to use GPU or CPU to run the algorithm.
    device = torch.device("cuda" if use_cuda else "cpu")

    train_kwargs = {'batch_size': args.batch_size}
    test_kwargs = {'batch_size': args.test_batch_size}
    if use_cuda:
        cuda_kwargs = {'num_workers': 1,
                       'pin_memory': True,
                       'shuffle': True}
        train_kwargs.update(cuda_kwargs)
        test_kwargs.update(cuda_kwargs)

    # Define the data preprocessing method.
    transform=transforms.Compose([
        transforms.ToTensor(),
        transforms.Normalize((0.1307,), (0.3081,))
        ])

    # Convert the raw MNIST dataset to a PyTorch MNIST dataset.
    convert_raw_mnist_dataset_to_pytorch_mnist_dataset(args.data_url)

    # Create a training dataset and a validation dataset.
    dataset1 = datasets.MNIST(args.data_url, train=True, download=False,
                       transform=transform)
    dataset2 = datasets.MNIST(args.data_url, train=False, download=False,
                       transform=transform)

    # Create iterators for the training dataset and the validation dataset.
    train_loader = torch.utils.data.DataLoader(dataset1, **train_kwargs)
    test_loader = torch.utils.data.DataLoader(dataset2, **test_kwargs)

    # Initialize the neural network model and copy the model to the compute
device.
    model = Net().to(device)
    # Define the training optimizer and learning rate for gradient descent
calculation.
    optimizer = optim.Adadelta(model.parameters(), lr=args.lr)
    scheduler = StepLR(optimizer, step_size=1, gamma=args.gamma)

    # Train the neural network and perform validation in each epoch.
    for epoch in range(1, args.epochs + 1):
        train(args, model, device, train_loader, optimizer, epoch)
        test(model, device, test_loader)
        scheduler.step()
```



```
    # Save the model and make it adapted to the ModelArts inference model package
specifications.
    if args.save_model:

        # Create the model directory in the path specified in train_url.
        model_path = os.path.join(args.train_url, 'model')
        os.makedirs(model_path, exist_ok = True)

        # Save the model to the model directory based on the ModelArts inference
model package specifications.
        torch.save(model.state_dict(), os.path.join(model_path, 'mnist_cnn.pt'))

        # Copy the inference code and configuration file to the model directory.
        the_path_of_current_file = os.path.dirname(__file__)
        shutil.copyfile(os.path.join(the_path_of_current_file, 'infer/
customize_service.py'), os.path.join(model_path, 'customize_service.py'))
        shutil.copyfile(os.path.join(the_path_of_current_file, 'infer/
config.json'), os.path.join(model_path, 'config.json'))

if __name__ == '__main__':
    main()
```

Cree el script de inferencia **customize_service.py** en el host local. El contenido es el siguiente:

```
import os
import log
import json

import torch.nn.functional as F
import torch.nn as nn
import torch
import torchvision.transforms as transforms

import numpy as np
from PIL import Image

from model_service.pytorch_model_service import PTServingBaseService

logger = log.getLogger(__name__)

# Define model preprocessing.
infer_transformation = transforms.Compose([
    transforms.Resize(28),
    transforms.CenterCrop(28),
    transforms.ToTensor(),
    transforms.Normalize((0.1307,), (0.3081,))
])

# Model inference service
class PTVisionService(PTServingBaseService):

    def __init__(self, model_name, model_path):
        # Call the constructor of the parent class.
        super(PTVisionService, self).__init__(model_name, model_path)

        # Call the customized function to load the model.
        self.model = Mnist(model_path)

         # Load labels.
        self.label = [0,1,2,3,4,5,6,7,8,9]
    # Receive the request data and convert it to the input format acceptable to
the model.
    def _preprocess(self, data):
        preprocessed_data = {}
        for k, v in data.items():
            input_batch = []
            for file_name, file_content in v.items():
                with Image.open(file_content) as image1:
```



```
                    # Gray processing
                    image1 = image1.convert("L")
                    if torch.cuda.is_available():
                        input_batch.append(infer_transformation(image1).cuda())
                    else:
                        input_batch.append(infer_transformation(image1))
            input_batch_var = torch.autograd.Variable(torch.stack(input_batch,
dim=0), volatile=True)
            print(input_batch_var.shape)
            preprocessed_data[k] = input_batch_var

        return preprocessed_data

    # Post-process the inference result to obtain the expected output format. The
result is the returned value.
    def _postprocess(self, data):
        results = []
        for k, v in data.items():
            result = torch.argmax(v[0])
            result = {k: self.label[result]}
            results.append(result)
        return results

    # Perform forward inference on the input data to obtain the inference result.
    def _inference(self, data):

        result = {}
        for k, v in data.items():
            result[k] = self.model(v)

        return result

# Define a network.
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, 3, 1)
        self.conv2 = nn.Conv2d(32, 64, 3, 1)
        self.dropout1 = nn.Dropout(0.25)
        self.dropout2 = nn.Dropout(0.5)
        self.fc1 = nn.Linear(9216, 128)
        self.fc2 = nn.Linear(128, 10)

    def forward(self, x):
        x = self.conv1(x)
        x = F.relu(x)
        x = self.conv2(x)
        x = F.relu(x)
        x = F.max_pool2d(x, 2)
        x = self.dropout1(x)
        x = torch.flatten(x, 1)
        x = self.fc1(x)
        x = F.relu(x)
        x = self.dropout2(x)
        x = self.fc2(x)
        output = F.log_softmax(x, dim=1)
        return output


def Mnist(model_path, **kwargs):
    # Generate a network.
    model = Net()

    # Load the model.
    if torch.cuda.is_available():
        device = torch.device('cuda')
        model.load_state_dict(torch.load(model_path, map_location="cuda:0"))
    else:
        device = torch.device('cpu')
```



```
        model.load_state_dict(torch.load(model_path, map_location=device))

    # CPU or GPU mapping
    model.to(device)

    # Turn the model to inference mode.
    model.eval()

    return model
```

Inferir el archivo de configuración **config.json** en el host local. El contenido es el siguiente:

```
{
    "model_algorithm": "image_classification",
    "model_type": "PyTorch",
    "runtime": "pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64"
}
```

Paso 3 Crear un bucket de OBS y cargar archivos a OBS

Cargue los datos, el archivo de código, el archivo de código de inferencia y el archivo de configuración de inferencia obtenidos en el paso anterior en un bucket de OBS. Al ejecutar un trabajo de entrenamiento en ModelArts, lea los archivos de datos y códigos del bucket del OBS.

1. Inicie sesión en la consola de gestión de OBS y cree un bucket y una carpeta de OBS. Para obtener más detalles, consulte **Creación de un bucket** y **Creación de una carpeta**.

```
{OBS bucket}                    # OBS bucket name, which is customizable,
for example, test-modelarts-xx
      -{OBS folder}          # OBS folder name, which is customizable, for
example, pytorch
          - mnist-data      # OBS folder, which is used to store the training
dataset. The folder name is customizable, for example, mnist-data.
          - mnist-code      # OBS folder, which is used to store training
script train.py. The folder name is customizable, for example, mnist-code.
              - infer       # OBS folder, which is used to store inference
script customize_service.py and configuration file config.json
          - mnist-output    # OBS folder, which is used to store trained
models. The folder name is customizable, for example, mnist-output.
```

⚠ ATENCIÓN

- La región donde reside el bucket de OBS creado debe ser la misma que donde se utiliza ModelArts. De lo contrario, el bucket de OBS estará disponible para entrenamiento. Para obtener más detalles, consulte **Comprobación de si el bucket de OBS y ModelArts se encuentran en la misma región**
- Al crear un bucket de OBS, no configure la clase de almacenamiento de archivos. De lo contrario, los modelos de entrenamiento fallarán.

2. Cargue el paquete de datos de MNIST obtenido en **Paso 1 Preparar los datos de entrenamiento** en OBS. Para obtener más detalles, consulte **Carga de un objeto**.

⚠ ATENCIÓN

- Al cargar datos en OBS, no los encripte. De lo contrario, el entrenamiento fallará.
- No es necesario que se decompongan. Cargue directamente paquetes comprimidos en el OBS.

3. Cargue el script de entrenamiento **train.py** en la carpeta **mnist-code**.
4. Cargue el script de inferencia **customize_service.py** y el archivo de configuración de inferencia **config.json** en la carpeta **infer**.

Paso 4 Crear un trabajo de entrenamiento

1. Inicie sesión en la consola de gestión de ModelArts y seleccione la misma región que el bucket del OBS.
2. En el panel de navegación de la izquierda, seleccione **Settings** y verifique si se ha configurado la autorización de acceso para la cuenta actual. Para obtener más detalles, consulte **Configuración de la autorización de acceso**. Si ha sido autorizado mediante claves de acceso, borre la autorización y configure la autorización de delegación.
3. En el panel de navegación, elija **Training Management** > **Training Jobs**. En la página **Training Jobs** que aparece, haga clic en **Create Training Job**.
4. Establezca los parámetros.
   - **Algorithm Type**: seleccione **Custom algorithm**.
   - **Boot Mode**: seleccione **Preset image** y, a continuación, seleccione **PyTorch** y **pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64** en las listas desplegables.
   - **Code Directory**: seleccione el directorio de códigos de OBS creado, por ejemplo, **/test-modelarts-xx/pytorch/mnist-code/** (reemplace **test-modelarts-xx** por su nombre de bucket de OBS).
   - **Boot File**: seleccione el script de entrenamiento **train.py** cargado en el directorio de códigos.
   - **Input**: Agregue una entrada y defina su nombre como **data_url**. Configure la ruta de acceso de datos al directorio de OBS. Por ejemplo, **/test-modelarts-xx/pytorch/mnist-data/** (reemplace **test-modelarts-xx** por el nombre del bucket de OBS).
   - **Output**: Agregue una salida y defina su nombre como **train_url**. Configure la ruta de acceso de datos al directorio de OBS. Por ejemplo, **/test-modelarts-xx/pytorch/mnist-output/** (reemplace **test-modelarts-xx** por el nombre del bucket de OBS). Establezca **Predownload** en **No**.
   - **Resource Type**: seleccione **GPU** y, a continuación, **GPU: 1*NVIDIA-V100(16GB) | CPU: 8 vCPUs 64GB** (ejemplo). Si existen especificaciones de GPU gratuitas, puede seleccionarlas para el entrenamiento.
   - Mantenga la configuración predeterminada para otros parámetros.

     NOTA

     El código de ejemplo se ejecuta en un solo nodo con una sola tarjeta. Si selecciona una variante con varias GPU, el entrenamiento fallará.
5. Haga clic en **Submit**, confirme la configuración de parámetros para el trabajo de entrenamiento y haga clic en **Yes**.

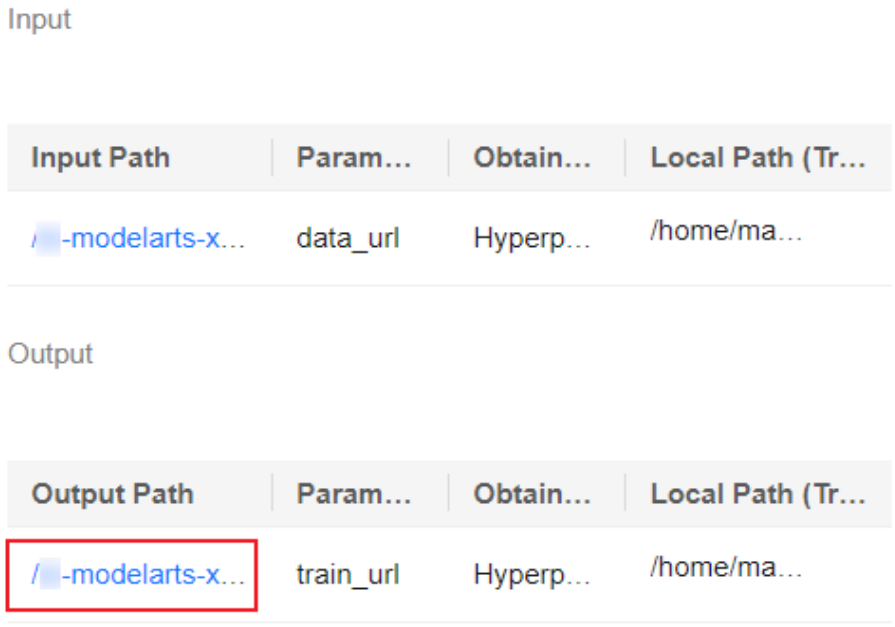
   El sistema vuelve automáticamente a la página **Training Jobs**. Cuando el estado del trabajo de entrenamiento cambia a **Completed**, se completa el entrenamiento de modelo.

   NOTA

   En este caso, el trabajo de entrenamiento durará unos 10 minutos.
6. Haga clic en el nombre del trabajo de entrenamiento. En la página de detalles del trabajo que aparece en pantalla, compruebe si hay mensajes de error en los registros. Si es así, el entrenamiento falló. Identifique la causa y localice la falla según los logs.

7. En la esquina inferior izquierda de la página de detalles del entrenamiento, haga clic en la ruta de salida del entrenamiento para ir a OBS, como se muestra en **Figura 4-4**. Luego, verifique si la carpeta **model** está disponible y si hay modelos entrenados en ella. Si no existe una carpeta **model** o un modelo entrenado, la información de entrenamiento puede estar incompleta. En este caso, cargue completamente los datos de entrenamiento y vuelva a entrenar el modelo.

Figura 4-4 Ruta de salida



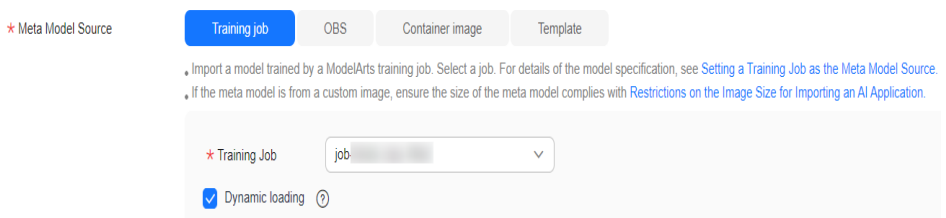
Paso 5: Desplegar el modelo para la inferencia

Una vez finalizado el entrenamiento del modelo, cree una aplicación de IA y despliegarla como un servicio en tiempo real.

1. Inicie sesión en la consola de gestión de ModelArts. En el panel de navegación de la izquierda, elija **AI Application Management** > **AI Applications**. En la página **My AI Applications**, haga clic en **Create**.
2. En la página **Create**, configure los parámetros y haga clic en **Create now**.

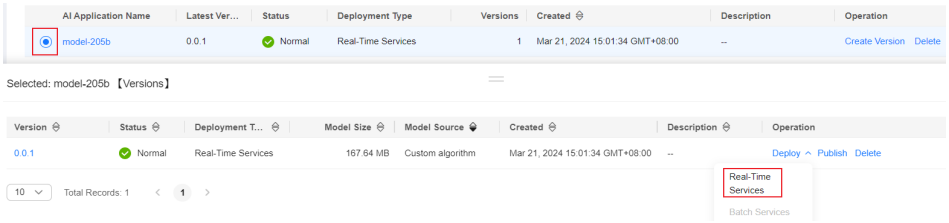
   Seleccione **Training Job** para **Meta Model Source**. En la lista desplegable, seleccione el trabajo de entrenamiento completado en **Paso 4 Crear un trabajo de entrenamiento** y seleccione **Dynamic loading**. Se configurarán automáticamente los valores de **AI Engine**.

Figura 4-5 Meta Model Source



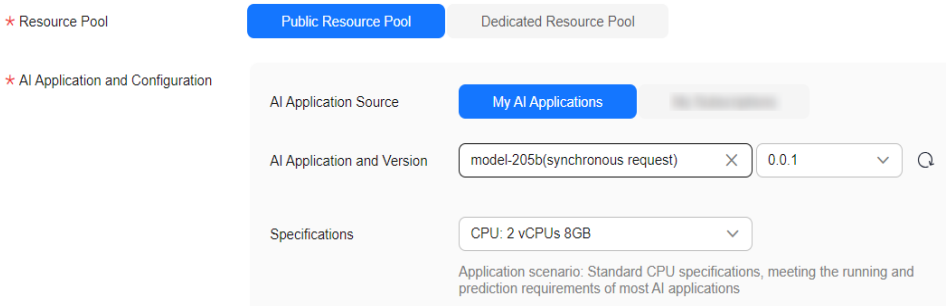
3. En la página **AI Applications**, si el estado de la aplicación cambia a **Normal**, se ha creado. Haga clic en el botón de opción situado a la izquierda del nombre de la aplicación de IA para mostrar la lista de versiones en la parte inferior de la página de lista y elija **Deploy** > **Real-Time Services** en la columna **Operation** para desplegar la aplicación de IA como un servicio en tiempo real.

Figura 4-6 Despliegue de un servicio en tiempo real



4. En la página **Deploy**, configure los parámetros y cree un servicio en tiempo real según se le solicite. En este ejemplo, utilice las especificaciones de CPU. Si existen especificaciones de CPU gratuitas, puede seleccionarlas para el despliegue. (Cada usuario puede desplegar solo un servicio en tiempo real de forma gratuita. Si ha desplegado uno, elimínelo primero antes de desplegar un nuevo de forma gratuita.)

Figura 4-7 Despliegue de un modelo



Después de enviar la solicitud de despliegue del servicio, el sistema cambia automáticamente a la página **Real-Time Services**. Cuando el estado del servicio cambia a **Running**, se despliegue ese servicio.

Paso 6 Realizar una predicción

1. En la página **Real-Time Services**, haga clic en el nombre del servicio en tiempo real. Aparecerá la página de detalles del servicio en tiempo real.
2. Haga clic en la ficha **Prediction**, configure **Request Type** como **multipart/form-data**, **Request Parameter** como **image**, haga clic en **Upload** para cargar una imagen de muestra y haga clic en **Predict**.

   Una vez finalizada la predicción, el resultado de la predicción se muestra en el panel **Test Result**. Según el resultado de la predicción, el dígito de la imagen es **2**.

   NOTA

   El MNIST utilizado en este caso es un conjunto de datos sencillo utilizado para la demostración, y sus algoritmos son también algoritmos de redes neuronales sencillos utilizados para la enseñanza. Los modelos generados con estos datos y algoritmos solo son aplicables a la enseñanza, pero no a la predicción compleja. La predicción solo es precisa si la imagen utilizada para la predicción es similar a la imagen del conjunto de datos de entrenamiento (caracteres blancos sobre fondo negro).

Figura 4-8 Ejemplo

Figura 4-9 Resultados de predicción

Paso 7 Lanzar recursos

Si ya no necesita usar este modelo y servicio en tiempo real, lance los recursos para detener la facturación.

- En la página **Real-Time Services**, localice la fila que contiene el servicio de destino y haga clic en **Stop** o **Delete** en la columna **Operation**.
- En la página **AI Applications** de **AI Application Management**, localice la fila que contiene el servicio de destino y haga clic en **Delete** en la columna **Operation**.
- En la página **Training Jobs**, haga clic en **Delete** en la columna **Operation** para eliminar el trabajo de entrenamiento finalizado.
- Vaya a OBS y elimine el bucket de OBS, las carpetas y los archivos utilizados en este ejemplo.

Preguntas frecuentes

- ¿Por qué un trabajo de entrenamiento siempre está en cola?

  Si el trabajo de entrenamiento está siempre en cola, los recursos seleccionados están limitados en el grupo de recursos y el trabajo debe estar en cola. En este caso, espere recursos. Para obtener más información, consulte **¿Por qué siempre está en cola un trabajo de entrenamiento?**.

- ¿Por qué no puedo encontrar mi bucket de OBS creado después de seleccionar una ruta de OBS en ModelArts?

  Asegúrese de que el bucket creado se encuentre en la misma región que ModelArts. Para obtener más detalles, consulte **Ruta de OBS incorrecta en ModelArts**.

4.2 Ejemplo: creación de una imagen personalizada para el entrenamiento (PyTorch + CPU/GPU)

En esta sección se describe cómo crear una imagen y utilizarla para entrenamiento en la plataforma de ModelArts. El motor de IA utilizado para el entrenamiento es PyTorch y los recursos son CPU o GPU.

> **NOTA**
>
> Esta sección solo se aplica a los trabajos de entrenamiento de la nueva versión.

Escenarios

En este ejemplo, cree una imagen personalizada escribiendo un Dockerfile en un host de Linux x86_64 que ejecute el sistema operativo Ubuntu 18.04.

Objetivo: crear e instalar imágenes de contenedor del siguiente software y utilizar las imágenes y las CPU/GPU para entrenamiento en ModelArts.

- ubuntu-18.04
- cuda-11.1
- python-3.7.13
- pytorch-1.8.1

Procedimiento

Antes de utilizar una imagen personalizada para crear un trabajo de entrenamiento, debe estar familiarizado con Docker y tener experiencia en desarrollo. El procedimiento detallado es el siguiente:

1. **Requisitos previos**
2. **Paso 1 Crear un bucket de OBS y una carpeta**
3. **Paso 2 Preparar el script de entrenamiento y cargarlo en OBS**
4. **Paso 3 Preparar un host**
5. **Paso 4 Crear una imagen personalizada**
6. **Paso 5 Cargar una imagen en SWR**
7. **Paso 6 Crear un trabajo de entrenamiento en ModelArts**

Requisitos previos

Ha creado un ID de Huawei y ha habilitado los servicios en Huawei Cloud. Además, la cuenta no está en mora ni congelada.

Paso 1 Crear un bucket de OBS y una carpeta

Cree un bucket y una carpeta en OBS para almacenar la muestra de conjunto de datos y el código de entrenamiento. **Tabla 4-1** enumera las carpetas que se crearán. En el ejemplo, el nombre del bucket y los nombres de las carpetas junto con los nombres reales.

Para obtener detalles sobre cómo crear un bucket de OBS y una carpeta, consulte **Creación de un bucket** y **Creación de una carpeta**.

Asegúrese de que el directorio de OBS que utiliza y ModelArts están en la misma región.

Tabla 4-1 Carpeta para crear

Nombre	Descripción
obs://test-modelarts/pytorch/demo-code/	Almacena el script de entrenamiento.
obs://test-modelarts/pytorch/log/	Almacena los archivos de log de entrenamiento.

Paso 2 Preparar el script de entrenamiento y cargarlo en OBS

Prepare el script de entrenamiento **pytorch-verification.py** y cárguelo en la carpeta **obs://test-modelarts/pytorch/demo-code/** del bucket del OBS.

El archivo **pytorch-verification.py** contiene la siguiente información:

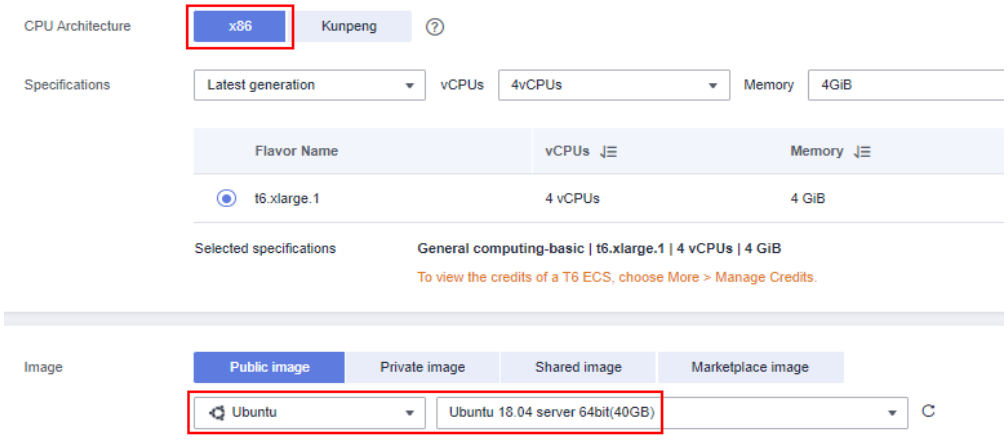
```
import torch
import torch.nn as nn

x = torch.randn(5, 3)
print(x)

available_dev = torch.device("cuda") if torch.cuda.is_available() else
torch.device("cpu")
y = torch.randn(5, 3).to(available_dev)
print(y)
```

Paso 3 Preparar un host

Obtener un servidor Linux x86_64 que ejecute Ubuntu 18.04. Un ECS o su PC local servirán.

Para obtener detalles sobre cómo comprar un ECS, consulte **Compra e inicio de sesión en un ECS de Linux**. Seleccione una imagen pública. Se recomienda una imagen de Ubuntu 18.04.

Figura 4-10 Creación de un ECS con una imagen pública (x86)



Paso 4 Crear una imagen personalizada

Cree una imagen de contenedor con las siguientes configuraciones y utilice la imagen para crear un trabajo de entrenamiento en ModelArts:

- ubuntu-18.04
- cuda-11.1
- python-3.7.13
- pytorch-1.8.1

En esta sección se describe cómo escribir un Dockerfile para crear una imagen personalizada.

1. Instale Docker.

   A continuación se utiliza Linux x86_64 OS como ejemplo para describir cómo obtener el paquete de instalación de Docker. Para obtener más detalles sobre cómo instalar Docker, consulte los **documentos oficiales de Docker**.

   ```
   curl -fsSL get.docker.com -o get-docker.sh
   sh get-docker.sh
   ```

   Si se ejecuta el comando **docker images**, Docker se ha instalado. Si es así, omita este paso.

2. Ejecute el siguiente comando para verificar la versión del Docker Engine:

   ```
   docker version | grep -A 1 Engine
   ```

   Se muestra la siguiente información:

   ```
   ...
   Engine:
     Version:          18.09.0
   ```

   NOTA

   Utilice el motor de Docker de la versión anterior o posterior para crear una imagen personalizada.

3. Cree una carpeta denominada **context**.

   ```
   mkdir -p context
   ```

4. Obtenga el archivo **pip.conf**. En este ejemplo, se utiliza el origen pip proporcionado por Huawei Mirrors, que es el siguiente:

   ```
   [global]
   index-url = https://repo.huaweicloud.com/repository/pypi/simple
   trusted-host = repo.huaweicloud.com
   timeout = 120
   ```

   NOTA

   En Huawei Mirrors **https://mirrors.huaweicloud.com/home**, busque **pypi** para obtener el archivo pip.conf.

5. Descargue los siguientes archivos .whl desde https://download.pytorch.org/whl/torch_stable.html:
   - torch-1.8.1+cu111-cp37-cp37m-linux_x86_64.whl
   - torchaudio-0.8.1-cp37-cp37m-linux_x86_64.whl
   - torchvision-0.9.1+cu111-cp37-cp37m-linux_x86_64.whl

   NOTA

   El código de URL del símbolo + es %2B. Cuando busque un archivo en el sitio web anterior, reemplace el símbolo + en el nombre del archivo por %2B.

   Por ejemplo, **torch-1.8.1%2Bcu111-cp37-cp37m-linux_x86_64.whl**.

6. Descargue el archivo de instalación Miniconda3-py37_4.12.0-Linux-x86_64.sh (Python 3.7.13) desde **https://repo.anaconda.com/miniconda/Miniconda3-py37_4.12.0-Linux-x86_64.sh**.
7. Almacene el archivo de origen de pip, el archivo torch*.whl y el archivo de instalación de Miniconda3 en la carpeta **context**, que es la siguiente:


```
context
├── Miniconda3-py37_4.12.0-Linux-x86_64.sh
├── pip.conf
├── torch-1.8.1+cu111-cp37-cp37m-linux_x86_64.whl
├── torchaudio-0.8.1-cp37-cp37m-linux_x86_64.whl
└── torchvision-0.9.1+cu111-cp37-cp37m-linux_x86_64.whl
```

8. Escribe la imagen del contenedor Dockerfile.

   Cree un archivo vacío denominado **Dockerfile** en la carpeta **context** y copie el siguiente contenido en el archivo:

```
# The host must be connected to the public network for creating a container
image.

# Base container image at https://github.com/NVIDIA/nvidia-docker/wiki/CUDA
#
# https://docs.docker.com/develop/develop-images/multistage-build/#use-multi-
stage-builds
# require Docker Engine >= 17.05
#
# builder stage
FROM nvidia/cuda:11.1.1-runtime-ubuntu18.04 AS builder

# The default user of the base container image is root.
# USER root

# Use the PyPI configuration provided by Huawei Mirrors.
RUN mkdir -p /root/.pip/
COPY pip.conf /root/.pip/pip.conf

# Copy the installation files to the /tmp directory in the base container
image.
COPY Miniconda3-py37_4.12.0-Linux-x86_64.sh /tmp
COPY torch-1.8.1+cu111-cp37-cp37m-linux_x86_64.whl /tmp
COPY torchvision-0.9.1+cu111-cp37-cp37m-linux_x86_64.whl /tmp
COPY torchaudio-0.8.1-cp37-cp37m-linux_x86_64.whl /tmp

# https://conda.io/projects/conda/en/latest/user-guide/install/
linux.html#installing-on-linux
# Install Miniconda3 to the /home/ma-user/miniconda3 directory of the base
container image.
RUN bash /tmp/Miniconda3-py37_4.12.0-Linux-x86_64.sh -b -p /home/ma-user/
miniconda3

# Install torch*.whl using the default Miniconda3 Python environment in /
home/ma-user/miniconda3/bin/pip.
RUN cd /tmp && \
    /home/ma-user/miniconda3/bin/pip install --no-cache-dir \
    /tmp/torch-1.8.1+cu111-cp37-cp37m-linux_x86_64.whl \
    /tmp/torchvision-0.9.1+cu111-cp37-cp37m-linux_x86_64.whl \
    /tmp/torchaudio-0.8.1-cp37-cp37m-linux_x86_64.whl

# Create the final container image.
FROM nvidia/cuda:11.1.1-runtime-ubuntu18.04

# Install vim and cURL in Huawei Mirrors.
RUN cp -a /etc/apt/sources.list /etc/apt/sources.list.bak && \
    sed -i "s@http://.*archive.ubuntu.com@http://
repo.huaweicloud.com@g" /etc/apt/sources.list && \
    sed -i "s@http://.*security.ubuntu.com@http://
repo.huaweicloud.com@g" /etc/apt/sources.list && \
    apt-get update && \
    apt-get install -y vim curl && \
    apt-get clean && \
    mv /etc/apt/sources.list.bak /etc/apt/sources.list

# Add user ma-user (UID = 1000, GID = 100).
# A user group whose GID is 100 of the base container image exists. User ma-
user can directly use it.
RUN useradd -m -d /home/ma-user -s /bin/bash -g 100 -u 1000 ma-user
```



```
# Copy the /home/ma-user/miniconda3 directory from the builder stage to the
directory with the same name in the current container image.
COPY --chown=ma-user:100 --from=builder /home/ma-user/miniconda3 /home/ma-
user/miniconda3

# Configure the preset environment variables of the container image.
# Set PYTHONUNBUFFERED to 1 to avoid log loss.
ENV PATH=$PATH:/home/ma-user/miniconda3/bin \
    PYTHONUNBUFFERED=1

# Set the default user and working directory of the container image.
USER ma-user
WORKDIR /home/ma-user
```

Para obtener detalles sobre cómo escribir un Dockerfile, consulte los **documentos oficiales de Docker**.

9. Verifique que se haya creado el Dockerfile. A continuación se muestra la carpeta **context**:

```
context
├── Dockerfile
├── Miniconda3-py37_4.12.0-Linux-x86_64.sh
├── pip.conf
├── torch-1.8.1+cu111-cp37-cp37m-linux_x86_64.whl
├── torchaudio-0.8.1-cp37-cp37m-linux_x86_64.whl
└── torchvision-0.9.1+cu111-cp37-cp37m-linux_x86_64.whl
```

10. Cree la imagen de contenedor. Ejecute el siguiente comando en el directorio donde se almacena el Dockerfile para crear la imagen de contenedor **pytorch:1.8.1-cuda11.1**:

```
docker build . -t pytorch:1.8.1-cuda11.1
```

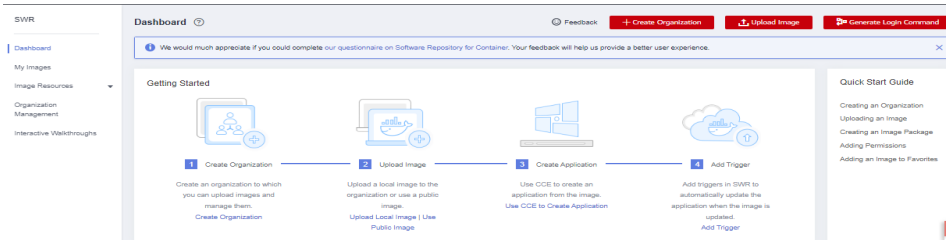
La siguiente información de log mostrada durante la creación de la imagen indica que la imagen se ha creado.

```
Successfully tagged pytorch:1.8.1-cuda11.1
```

Paso 5 Cargar una imagen en SWR

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

Figura 4-11 Consola de SWR

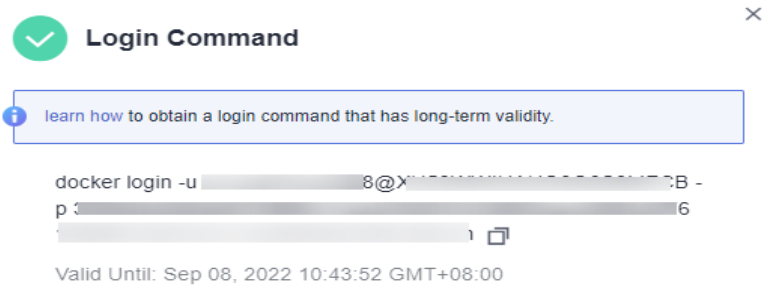


2. Haga clic en **Create Organization** en la esquina superior derecha e introduzca un nombre de organización para crear una organización. Personalice el nombre de la organización. Sustituya el nombre de la organización **deep-learning** en comandos posteriores con el nombre real de la organización.

Figura 4-12 Creación de una organización

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

Figura 4-13 Comando de acceso



4. Inicie sesión en el entorno local como usuario **root** e ingrese el comando de inicio de sesión.
5. Cargue la imagen en SWR.
   a. Ejecute el siguiente comando para etiquetar la imagen cargada:
   ```
   #Replace the region and domain information with the actual values, and replace the organization name deep-learning with your custom value.
   sudo docker tag pytorch:1.8.1-cuda11.1 swr.{region-id}.{domain}/deep-learning/pytorch:1.8.1-cuda11.1
   ```
   b. Ejecute el siguiente comando para subir la imagen:
   ```
   #Replace the region and domain information with the actual values, and replace the organization name deep-learning with your custom value.
   sudo docker push swr.{region-id}.{domain}/deep-learning/pytorch:1.8.1-cuda11.1
   ```
6. 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.

Paso 6 Crear un trabajo de entrenamiento en ModelArts

1. Inicie sesión en la consola de gestión de ModelArts y compruebe si se ha configurado la autorización de acceso para su cuenta. Para obtener más detalles, consulte **Configuración de autorización de delegación**. Si se le ha autorizado mediante claves de acceso, borre la autorización y configure la autorización de delegación.
2. En el panel de navegación, seleccione **Training Management** > **Training Jobs**. La lista de trabajos de entrenamiento se muestra de forma predeterminada.

3. En la página **Create Training Job**, configure los parámetros necesarios y haga clic en **Submit**.
   - **Created By**: **Custom algorithms**
   - **Boot Mode**: **Custom images**
   - Ruta de la imagen: imagen creada en **Paso 5 Cargar una imagen en SWR**.
   - **Code Directory**: directorio donde se almacena el archivo de script de arranque en OBS. Por ejemplo, **obs://test-modelarts/pytorch/demo-code/**. El código de entrenamiento se descarga automáticamente en el directorio **${MA_JOB_DIR}/demo-code** del contenedor de entrenamiento. **demo-code** (personalizable) es el directorio de último nivel de la ruta del OBS.
   - **Boot Command**: **/home/ma-user/miniconda3/bin/python ${MA_JOB_DIR}/demo-code/pytorch-verification.py**. **demo-code** (personalizable) es el directorio de último nivel de la ruta del OBS.
   - **Resource Pool**: **Public resource pools**
   - **Resource Type**: seleccione **CPU** o **GPU**.
   - **Persistent Log Saving**: habilitado
   - **Job Log Path**: Establezca este parámetro en la ruta de OBS para almacenar logs de entrenamiento, por ejemplo, **obs://test-modelarts/pytorch/log/**.
4. Verifique los parámetros del trabajo de entrenamiento y haga clic en **Submit**.
5. Espere hasta que finalice el trabajo de entrenamiento.

   Después de crear un trabajo de entrenamiento, las operaciones como la descarga de imágenes de contenedor, la descarga de directorios de código y la ejecución de comandos de arranque se realizan automáticamente en el backend. Por lo general, la duración del entrenamiento oscila entre decenas de minutos y varias horas, dependiendo del procedimiento de entrenamiento y de los recursos seleccionados. Una vez ejecutado el trabajo de entrenamiento, se muestra un log similar al siguiente.

   Figura 4-14 Ejecutar logs de trabajos de entrenamiento con especificaciones de GPU

```
tensor([[-0.4181,  0.8150, -0.2581],
        [-0.6062,  0.5347,  0.1890],
        [ 0.5751,  1.2730, -0.3907],
        [ 0.4812, -0.4064, -0.2753],
        [ 1.0377, -1.1248,  1.2977]])
tensor([[-0.7440, -0.8577, -0.2340],
        [ 0.9569,  0.5516, -1.3350],
        [-1.2878, -0.2791,  0.3486],
        [-1.0997,  0.7627, -0.3188],
        [-1.0865, -1.2626, -0.5900]], device='cuda:0')

```


Figura 4-15 Run logs of training jobs with CPU specifications

```
tensor([[ 0.8945, -0.6946,  0.3807],
        [ 0.6665,  0.3133,  0.8285],
        [-0.5353, -0.1730, -0.5419],
        [ 0.4870,  0.5183,  0.2505],
        [ 0.2679, -0.4996,  0.7919]])
tensor([[ 0.9692,  0.4652,  0.5659],
        [ 2.2032,  1.4157, -0.1755],
        [-0.6296,  0.5466,  0.6994],
        [ 0.2353, -0.0089, -1.9546],
        [ 0.9319,  1.1781, -0.4587]])

```

4.3 Ejemplo: creación de una imagen personalizada para entrenamiento (MPI + CPU/GPU)

En esta sección se describe cómo crear una imagen y utilizarla para entrenamiento en la plataforma de ModelArts. El motor de IA utilizado para el entrenamiento es MPI y los recursos son CPU o GPU.

NOTA

Esta sección solo se aplica a los trabajos de entrenamiento de la nueva versión.

Escenarios

En este ejemplo, cree una imagen personalizada escribiendo un Dockerfile en un host de Linux x86_64 que ejecute el sistema operativo Ubuntu 18.04.

Objetivo: crear e instalar imágenes de contenedor del siguiente software y utilizar las imágenes y las CPU/GPU para entrenamiento en ModelArts.

- ubuntu-18.04
- cuda-11.1
- python-3.7.13
- openmpi-3.0.0

Procedimiento

Antes de usar una imagen personalizada para crear un trabajo de entrenamiento, familiarícese con Docker y tenga experiencia en desarrollo. El procedimiento detallado es el siguiente:

1. **Requisitos previos**
2. **Paso 1 Crear un bucket de OBS y una carpeta**

3. **Paso 2 Preparar los archivos y cargarlos en OBS**
4. **Paso 3 Preparar un servidor de imágenes**
5. **Paso 4 Crear una imagen personalizada**
6. **Paso 5 Cargar una imagen en SWR**
7. **Paso 6 Crear un trabajo de entrenamiento en ModelArts**

Requisitos previos

Ha creado un ID de Huawei y ha habilitado los servicios en Huawei Cloud. Además, la cuenta no está en mora ni congelada.

Paso 1 Crear un bucket de OBS y una carpeta

Cree un bucket y una carpeta en OBS para almacenar la muestra de conjunto de datos y el código de entrenamiento. **Tabla 4-2** enumera las carpetas que se crearán. En el ejemplo, el nombre del bucket y los nombres de las carpetas junto con los nombres reales.

Para obtener detalles sobre cómo crear un bucket de OBS y una carpeta, consulte **Creación de un bucket** y **Creación de una carpeta**.

Asegúrese de que el directorio de OBS que utiliza y ModelArts están en la misma región.

Tabla 4-2 Carpeta para crear

Nombre	Descripción
obs://test-modelarts/mpi/demo-code/	Almacena el script de arranque de MPI y el archivo de script de entrenamiento.
obs://test-modelarts/mpi/log/	Almacena los archivos de log de entrenamiento.

Paso 2 Preparar los archivos y cargarlos en OBS

Prepare el script de inicio de MPI **run_mpi.sh** y el script de entrenamiento **mpi-verification.py** y cárguelos en la carpeta **obs://test-modelarts/mpi/demo-code/** del bucket de OBS.

- El contenido del script de inicio de MPI **run_mpi.sh** es el siguiente:

```
#!/bin/bash
MY_HOME=/home/ma-user

MY_SSHD_PORT=${MY_SSHD_PORT:-"38888"}

MY_TASK_INDEX=${MA_TASK_INDEX:-${VC_TASK_INDEX:-${VK_TASK_INDEX}}}

MY_MPI_SLOTS=${MY_MPI_SLOTS:-"${MA_NUM_GPUS}"}

MY_MPI_TUNE_FILE="${MY_HOME}/env_for_user_process"

if [ -z ${MY_MPI_SLOTS} ]; then
    echo "[run_mpi] MY_MPI_SLOTS is empty, set it be 1"
    MY_MPI_SLOTS="1"
fi

printf "MY_HOME: ${MY_HOME}\nMY_SSHD_PORT: ${MY_SSHD_PORT}
```



```
\nMY_MPI_BTL_TCP_IF: ${MY_MPI_BTL_TCP_IF}\nMY_TASK_INDEX: ${MY_TASK_INDEX}
\nMY_MPI_SLOTS: ${MY_MPI_SLOTS}\n"

env | grep -E '^MA_|^SHARED_|^S3_|^PATH|^VC_WORKER_|^SCC|^CRED' | grep -v '=
$' > ${MY_MPI_TUNE_FILE}
# add -x to each line
sed -i 's/^/-x /' ${MY_MPI_TUNE_FILE}

sed -i "s|{{MY_SSHD_PORT}}|${MY_SSHD_PORT}|g" ${MY_HOME}/etc/ssh/sshd_config

# start sshd service
bash -c "$(which sshd) -f ${MY_HOME}/etc/ssh/sshd_config"

# confirm the sshd is up
netstat -anp | grep LIS | grep ${MY_SSHD_PORT}

if [ $MY_TASK_INDEX -eq 0 ]; then
    # generate the hostfile of mpi
    for ((i=0; i<$MA_NUM_HOSTS; i++))
    do
        eval hostname=${MA_VJ_NAME}-${MA_TASK_NAME}-${i}.${MA_VJ_NAME}
        echo "[run_mpi] hostname: ${hostname}"

        ip=""
        while [ -z "$ip" ]; do
            ip=$(ping -c 1 ${hostname} | grep "PING" | sed -E 's/PING .* .
([0-9.]+). .*/\1/g')
            sleep 1
        done
        echo "[run_mpi] resolved ip: ${ip}"

        # test the sshd is up
        while :
        do
            if [ cat < /dev/null >/dev/tcp/${ip}/${MY_SSHD_PORT} ]; then
                break
            fi
            sleep 1
        done

        echo "[run_mpi] the sshd of ip ${ip} is up"

        echo "${ip} slots=$MY_MPI_SLOTS" >> ${MY_HOME}/hostfile
    done

    printf "[run_mpi] hostfile:\n`cat ${MY_HOME}/hostfile`\n"
fi

RET_CODE=0

if [ $MY_TASK_INDEX -eq 0 ]; then

    echo "[run_mpi] start exec command time: "$(date +"%Y-%m-%d-%H:%M:%S")

    np=$(( ${MA_NUM_HOSTS} * ${MY_MPI_SLOTS} ))

    echo "[run_mpi] command: mpirun -np ${np} -hostfile ${MY_HOME}/hostfile -
mca plm_rsh_args \"-p ${MY_SSHD_PORT}\" -tune ${MY_MPI_TUNE_FILE} ... $@"

    # execute mpirun at worker-0
    # mpirun
    mpirun \
        -np ${np} \
        -hostfile ${MY_HOME}/hostfile \
        -mca plm_rsh_args "-p ${MY_SSHD_PORT}" \
        -tune ${MY_MPI_TUNE_FILE} \
        -bind-to none -map-by slot \
        -x NCCL_DEBUG -x NCCL_SOCKET_IFNAME -x NCCL_IB_HCA -x NCCL_IB_TIMEOUT
-x NCCL_IB_GID_INDEX -x NCCL_IB_TC \
```



```
        -x HOROVOD_MPI_THREADS_DISABLE=1 \
        -x PATH -x LD_LIBRARY_PATH \
        -mca pml ob1 -mca btl ^openib -mca plm_rsh_no_tree_spawn true \
        "$@"

    RET_CODE=$?

    if [ $RET_CODE -ne 0 ]; then
        echo "[run_mpi] exec command failed, exited with $RET_CODE"
    else
        echo "[run_mpi] exec command successfully, exited with $RET_CODE"
    fi

    # stop 1...N worker by killing the sleep proc
    sed -i '1d' ${MY_HOME}/hostfile
    if [ `cat ${MY_HOME}/hostfile | wc -l` -ne 0 ]; then
        echo "[run_mpi] stop 1 to (N - 1) worker by killing the sleep proc"

        sed -i 's/${MY_MPI_SLOTS}/1/g' ${MY_HOME}/hostfile
        printf "[run_mpi] hostfile:\n`cat ${MY_HOME}/hostfile`\n"

        mpirun \
        --hostfile ${MY_HOME}/hostfile \
        --mca plm_rsh_args "-p ${MY_SSHD_PORT}" \
        -x PATH -x LD_LIBRARY_PATH \
        pkill sleep \
        > /dev/null 2>&1
    fi

    echo "[run_mpi] exit time: "$(date +"%Y-%m-%d-%H:%M:%S")
else
    echo "[run_mpi] the training log is in worker-0"
    sleep 365d
    echo "[run_mpi] exit time: "$(date +"%Y-%m-%d-%H:%M:%S")
fi

exit $RET_CODE
```

> **NOTA**
>
> El script **run_mpi.sh** utiliza finales de línea LF. Si se utilizan finales de línea CRLF, la ejecución del trabajo de entrenamiento fallará y se mostrará el error "$'\r': command not found" en los logs.

- El contenido del script de entrenamiento **mpi-verification.py** es el siguiente:

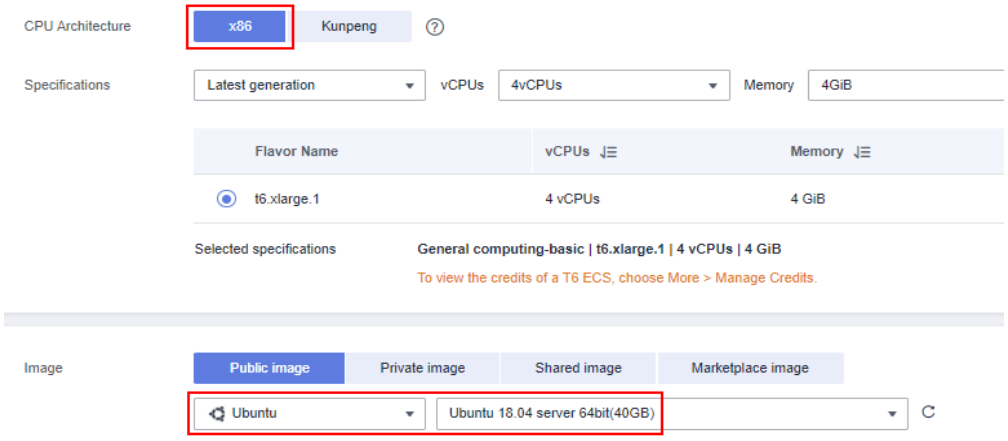
```
import os
import socket

if __name__ == '__main__':
    print(socket.gethostname())

    # https://www.open-mpi.org/faq/?category=running#mpi-environmental-
variables
    print('OMPI_COMM_WORLD_SIZE: ' + os.environ['OMPI_COMM_WORLD_SIZE'])
    print('OMPI_COMM_WORLD_RANK: ' + os.environ['OMPI_COMM_WORLD_RANK'])
    print('OMPI_COMM_WORLD_LOCAL_RANK: ' +
os.environ['OMPI_COMM_WORLD_LOCAL_RANK'])
```

Paso 3 Preparar un servidor de imágenes

Obtener un servidor Linux x86_64 que ejecute Ubuntu 18.04. Un ECS o su PC local servirán.

Para obtener detalles sobre cómo comprar un ECS, consulte **Compra e inicio de sesión en un ECS de Linux**. Seleccione una imagen pública. Se recomienda una imagen de Ubuntu 18.04.

Figura 4-16 Creación de un ECS con una imagen pública (x86)



Paso 4 Crear una imagen personalizada

Objetivo: crear e instalar imágenes de contenedor del siguiente software y utilizar el servicio de entrenamiento de ModelArts para ejecutarlas.

- ubuntu-18.04
- cuda-11.1
- python-3.7.13
- openmpi-3.0.0

A continuación se describe cómo crear una imagen personalizada escribiendo un Dockerfile.

1. Instale Docker.

   A continuación se utiliza Linux x86_64 OS como ejemplo para describir cómo obtener un paquete de instalación de Docker. Para obtener más detalles, consulte los **documentos oficiales de Docker**. Ejecute los siguientes comandos para instalar Docker:

   ```
   curl -fsSL get.docker.com -o get-docker.sh
   sh get-docker.sh
   ```

   Si se ejecuta el comando **docker images**, Docker se ha instalado. Si es así, omita este paso.

2. Verifique la versión del motor de Docker. Ejecute el siguiente comando:

   ```
   docker version | grep -A 1 Engine
   ```

   Se muestra la siguiente información:

   ```
   Engine:
    Version:          18.09.0
   ```

   > **NOTA**
   >
   > Se recomienda utilizar el Docker Engine de esta versión o posterior para crear una imagen personalizada.

3. Cree una carpeta denominada **context**.

   ```
   mkdir -p context
   ```

4. Descargue el archivo de instalación de Miniconda3.

   Descargue el archivo de instalación de Miniconda3 py37 4.12.0 (Python 3.7.13) desde **https://repo.anaconda.com/miniconda/Miniconda3-py37_4.12.0-Linux-x86_64.sh**.

5. Descargue el archivo de instalación de openmpi 3.0.0.

Descargue el archivo de openmpi 3.0.0 editado con Horovod v0.22.1 desde **https://github.com/horovod/horovod/files/1596799/openmpi-3.0.0-bin.tar.gz**.

6. Guarde los archivos de Miniconda3 y de openmpi 3.0.0 en la carpeta **context**. A continuación se muestra la carpeta **context**:

```
context
├── Miniconda3-py37_4.12.0-Linux-x86_64.sh
└── openmpi-3.0.0-bin.tar.gz
```

7. Escriba el Dockerfile de la imagen de contenedor.

   Cree un archivo vacío denominado **Dockerfile** en la carpeta **context** y escriba el siguiente contenido en el archivo:

```
# The host must be connected to the public network for creating a container
image.

# Basic container image at https://github.com/NVIDIA/nvidia-docker/wiki/CUDA
#
# https://docs.docker.com/develop/develop-images/multistage-build/#use-multi-
stage-builds
# require Docker Engine >= 17.05
#
# builder stage
FROM nvidia/cuda:11.1.1-runtime-ubuntu18.04 AS builder

# The default user of the basic container image is root.
# USER root

# Copy the Miniconda3 (Python 3.7.13) installation files to the /tmp
directory of the basic container image.
COPY Miniconda3-py37_4.12.0-Linux-x86_64.sh /tmp

# Install Miniconda3 to the /home/ma-user/miniconda3 directory of the basic
container image.
# https://conda.io/projects/conda/en/latest/user-guide/install/
linux.html#installing-on-linux
RUN bash /tmp/Miniconda3-py37_4.12.0-Linux-x86_64.sh -b -p /home/ma-user/
miniconda3

# Create the final container image.
FROM nvidia/cuda:11.1.1-runtime-ubuntu18.04

# Install vim, cURL, net-tools, and the SSH tool in Huawei Mirrors.
RUN cp -a /etc/apt/sources.list /etc/apt/sources.list.bak && \
    sed -i "s@http://.*archive.ubuntu.com@http://
repo.huaweicloud.com@g" /etc/apt/sources.list && \
    sed -i "s@http://.*security.ubuntu.com@http://
repo.huaweicloud.com@g" /etc/apt/sources.list && \
    echo > /etc/apt/apt.conf.d/00skip-verify-peer.conf "Acquire
{ https::Verify-Peer false }" && \
    apt-get update && \
    apt-get install -y vim curl net-tools iputils-ping \
    openssh-client openssh-server && \
    ssh -V && \
    mkdir -p /run/sshd && \
    apt-get clean && \
    mv /etc/apt/sources.list.bak /etc/apt/sources.list && \
    rm /etc/apt/apt.conf.d/00skip-verify-peer.conf

# Install the Open MPI 3.0.0 file written using Horovod v0.22.1.
# https://github.com/horovod/horovod/blob/v0.22.1/docker/horovod/Dockerfile
# https://github.com/horovod/horovod/files/1596799/openmpi-3.0.0-bin.tar.gz
COPY openmpi-3.0.0-bin.tar.gz /tmp
RUN cd /usr/local && \
    tar -zxf /tmp/openmpi-3.0.0-bin.tar.gz && \
    ldconfig && \
    mpirun --version

# Add user ma-user (UID = 1000, GID = 100).
```



```
# A user group whose GID is 100 of the basic container image exists. User ma-
user can directly use it.
RUN useradd -m -d /home/ma-user -s /bin/bash -g 100 -u 1000 ma-user

# Copy the /home/ma-user/miniconda3 directory from the builder stage to the
directory with the same name in the current container image.
COPY --chown=ma-user:100 --from=builder /home/ma-user/miniconda3 /home/ma-
user/miniconda3

# Configure the preset environment variables of the container image.
# Set PYTHONUNBUFFERED to 1 to avoid log loss.
ENV PATH=$PATH:/home/ma-user/miniconda3/bin \
    PYTHONUNBUFFERED=1

# Set the default user and working directory of the container image.
USER ma-user
WORKDIR /home/ma-user

# Configure sshd to support SSH password-free login.
RUN MA_HOME=/home/ma-user && \
    # setup sshd dir
    mkdir -p ${MA_HOME}/etc && \
    ssh-keygen -f ${MA_HOME}/etc/ssh_host_rsa_key -N '' -t rsa  && \
    mkdir -p ${MA_HOME}/etc/ssh ${MA_HOME}/var/run  && \
    # setup sshd config (listen at {{MY_SSHD_PORT}} port)
    echo "Port {{MY_SSHD_PORT}}\n\
HostKey ${MA_HOME}/etc/ssh_host_rsa_key\n\
AuthorizedKeysFile ${MA_HOME}/.ssh/authorized_keys\n\
PidFile ${MA_HOME}/var/run/sshd.pid\n\
StrictModes no\n\
UsePAM no" > ${MA_HOME}/etc/ssh/sshd_config && \
    # generate ssh key
    ssh-keygen -t rsa -f ${MA_HOME}/.ssh/id_rsa -P '' && \
    cat ${MA_HOME}/.ssh/id_rsa.pub >> ${MA_HOME}/.ssh/authorized_keys && \
    # disable ssh host key checking for all hosts
    echo "Host *\n\
  StrictHostKeyChecking no" > ${MA_HOME}/.ssh/config
```

Para obtener detalles sobre cómo escribir un Dockerfile, consulte los **Documentos oficiales de Docker**.

8. Verifique que se haya creado el Dockerfile. A continuación se muestra la carpeta **context**:

```
context
├── Dockerfile
├── Miniconda3-py37_4.12.0-Linux-x86_64.sh
└── openmpi-3.0.0-bin.tar.gz
```

9. Cree la imagen de contenedor. Ejecute el siguiente comando en el directorio donde se almacena Dockerfile para crear la imagen de contenedor **mpi:3.0.0-cuda11.1**:

```
docker build . -t mpi:3.0.0-cuda11.1
```

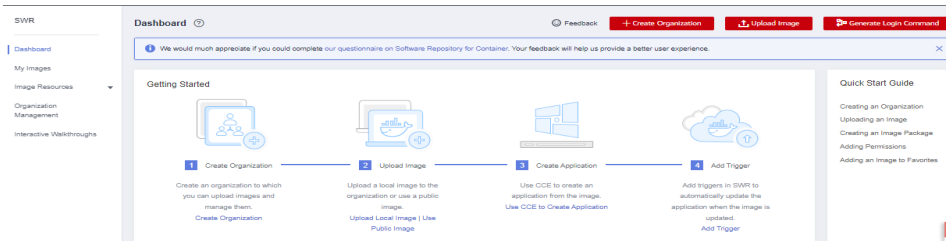
La siguiente información de log mostrada durante la creación de la imagen indica que la imagen se ha creado.

```
naming to docker.io/library/mpi:3.0.0-cuda11.1
```

Paso 5 Cargar una imagen en SWR

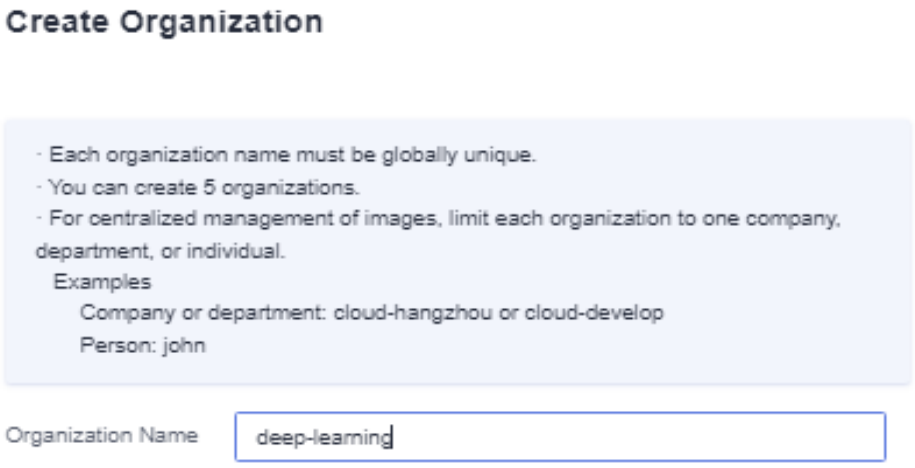
1. Inicie sesión en la consola de SWR y seleccione la región de destino.

Figura 4-17 Consola de SWR



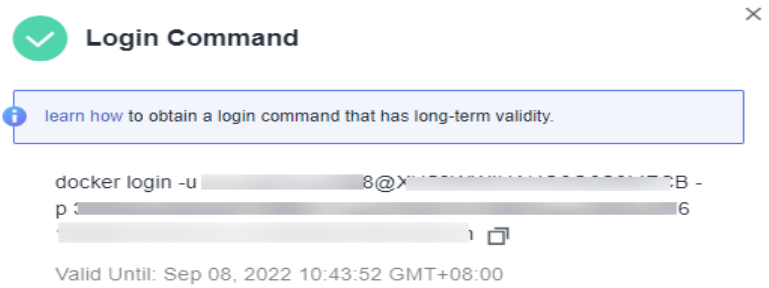
2. Haga clic en **Create Organization** en la esquina superior derecha e introduzca un nombre de organización para crear una organización. Personalice el nombre de la organización. Sustituya el nombre de la organización **deep-learning** en comandos posteriores con el nombre real de la organización.

Figura 4-18 Creación de una organización



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

Figura 4-19 Comando de acceso



4. Inicie sesión en el entorno local como usuario **root** e ingrese el comando de inicio de sesión.
5. Cargue la imagen en SWR.
   a. Ejecute el siguiente comando para etiquetar la imagen cargada:

```
#Replace the region and domain information with the actual values, and replace the organization name deep-learning with your custom value.
sudo docker tag mpi:3.0.0-cuda11.1 swr.cn-north-4.myhuaweicloud.com/deep-learning/mpi:3.0.0-cuda11.1
```


b. Ejecute el siguiente comando para subir la imagen:

```
#Replace the region and domain information with the actual values, and
replace the organization name deep-learning with your custom value.
sudo docker push swr.cn-north-4.myhuaweicloud.com/deep-learning/
mpi:3.0.0-cuda11.1
```

6. Después de cargar la imagen, seleccione **My Images** en el panel de navegación izquierdo de la consola de SWR para ver las imágenes personalizadas cargadas.

   swr.cn-north-4.myhuaweicloud.com/deep-learning/mpi:3.0.0-cuda11.1 es la dirección URL de SWR de la imagen personalizada.

Paso 6 Crear un trabajo de entrenamiento en ModelArts

1. Inicie sesión en la consola de gestión de ModelArts y compruebe si se ha configurado la autorización de acceso para su cuenta. Para obtener más detalles, consulte **Configuración de autorización de delegación**. Si se le ha autorizado mediante claves de acceso, borre la autorización y configure la autorización de delegación.
2. Inicie sesión en la consola de gestión de ModelArts. En el panel de navegación izquierdo, seleccione **Training Management** > **Training Jobs (New)**.
3. En la página **Create Training Job**, configure los parámetros y haga clic en **Submit**.
   - **Created By**: **Custom algorithms**
   - **Boot Mode**: **Custom images**
   - Ruta de acceso a la imagen: **swr.cn-north-4.myhuaweicloud.com/deep-learning/mpi:3.0.0-cuda11.1**
   - **Code Directory**: ruta de acceso de OBS al script de inicio, por ejemplo, **obs://test-modelarts/mpi/demo-code/**.
   - **Boot Command**: **bash ${MA_JOB_DIR}/demo-code/run_mpi.sh python ${MA_JOB_DIR}/demo-code/mpi-verification.py**
   - **Environment Variable**: Agregue **MY_SSHD_PORT = 38888**.
   - **Resource Pool**: **Public resource pools**
   - **Resource Type**: seleccione **GPU**.
   - **Compute Nodes**: Introduzca **1** o **2**.
   - **Persistent Log Saving**: habilitado
   - **Job Log Path**: Establezca este parámetro en la ruta de OBS para almacenar logs de entrenamiento, por ejemplo, **obs://test-modelarts/mpi/log/**.
4. Verifique los parámetros del trabajo de entrenamiento y haga clic en **Submit**.
5. Espere hasta que finalice el trabajo de entrenamiento.

   Después de crear un trabajo de entrenamiento, las operaciones como la descarga de imágenes de contenedor, la descarga de directorios de código y la ejecución de comandos de arranque se realizan automáticamente en el backend. Por lo general, la duración del entrenamiento oscila entre decenas de minutos y varias horas, dependiendo del procedimiento de entrenamiento y de los recursos seleccionados. Una vez ejecutado el trabajo de entrenamiento, se muestra un log similar al siguiente.

Figura 4-20 Ejecutar logs de worker-0 con un nodo de cómputo y especificaciones de GPU

```
MY_HOME: /home/ma-user
MY_SSHD_PORT: 38888
MY_MPI_BTL_TCP_IF: eth0,bond0
MY_TASK_INDEX: 0
MY_MPI_SLOTS: 1
tcp        0      0 0.0.0.0:38888           0.0.0.0:*               LISTEN      60/sshd
tcp6       0      0 :::38888                :::*                    LISTEN      60/sshd
172.16.0.122 slots=1
modelarts-job-8cf8a682-21cb-4d73-9bb3-789cecdc458b-worker-0
OMPI_COMM_WORLD_SIZE: 1
OMPI_COMM_WORLD_RANK: 0
OMPI_COMM_WORLD_LOCAL_RANK: 0
```

Configure **Compute Nodes** como **2** y ejecute el trabajo de entrenamiento. **Figura 4-21** y **Figura 4-22** muestran la información del log.

Figura 4-21 Ejecutar logs de worker-0 con dos nodos de cómputo y especificaciones de GPU

```
MY_HOME: /home/ma-user
MY_SSHD_PORT: 38888
MY_MPI_BTL_TCP_IF: eth0,bond0
MY_TASK_INDEX: 0
MY_MPI_SLOTS: 1
tcp        0      0 0.0.0.0:38888           0.0.0.0:*               LISTEN      61/sshd
tcp6       0      0 :::38888                :::*                    LISTEN      61/sshd
172.16.0.39 slots=1
172.16.0.123 slots=1
Warning: Permanently added '[172.16.0.123]:38888' (RSA) to the list of known hosts.
modelarts-job-31732752-6857-4e33-96ff-7a28afae26fb-worker-0
OMPI_COMM_WORLD_SIZE: 2
OMPI_COMM_WORLD_RANK: 0
OMPI_COMM_WORLD_LOCAL_RANK: 0
modelarts-job-31732752-6857-4e33-96ff-7a28afae26fb-worker-1
OMPI_COMM_WORLD_SIZE: 2
OMPI_COMM_WORLD_RANK: 1
OMPI_COMM_WORLD_LOCAL_RANK: 0
```

Figura 4-22 Ejecutar logs de worker-1 con dos nodos de cómputo y especificaciones de GPU

```
MY_HOME: /home/ma-user
MY_SSHD_PORT: 38888
MY_MPI_BTL_TCP_IF: eth0,bond0
MY_TASK_INDEX: 1
MY_MPI_SLOTS: 1
tcp        0      0 0.0.0.0:38888           0.0.0.0:*               LISTEN      62/sshd
tcp6       0      0 :::38888                :::*                    LISTEN      62/sshd
/home/ma-user/modelarts/user-job-dir/[redacted]e/run_mpi.sh: line 109:    66 Terminated              sleep 365d
```

4.4 Ejemplo: creación de una imagen personalizada para entrenamiento (Horovod-PyTorch y GPU)

En esta sección se describe cómo crear una imagen y utilizarla para los entrenamientos en ModelArts. El motor de IA utilizado en la imagen es Horovod 0.22.1 + PyTorch 1.8.1 y los recursos utilizados para el entrenamiento son GPU.

> **NOTA**
>
> Esta sección solo se aplica a los trabajos de entrenamiento de la nueva versión.

Escenario

En este ejemplo, escriba un Dockerfile para crear una imagen personalizada en un servidor Linux x86_64 que ejecute Ubuntu 18.04.

Objetivo: crear e instalar imágenes de contenedor del siguiente software y utilizar las imágenes y las CPU/GPU para entrenamiento en ModelArts.

- ubuntu-18.04
- cuda-11.1
- python-3.7.13
- mlnx ofed-5.4
- pytorch-1.8.1
- horovod-0.22.1

Procedimiento

Antes de utilizar una imagen personalizada para crear un trabajo de entrenamiento, debe estar familiarizado con Docker y tener experiencia en desarrollo.

1. **Requisitos previos**
2. **Paso 1 Crear un bucket de OBS y una carpeta**
3. **Paso 2 Preparar el script de entrenamiento y cargarlo en OBS**
4. **Paso 3 Preparar un servidor**
5. **Paso 4 Crear una imagen personalizada**
6. **Paso 5 Cargar la imagen en SWR**
7. **Paso 6 Crear un trabajo de entrenamiento en ModelArts**

Requisitos previos

Ha creado una cuenta en Huawei Cloud. La cuenta no está en mora ni congelada.

Paso 1 Crear un bucket de OBS y una carpeta

Cree un bucket y una carpeta en OBS para almacenar la muestra de conjunto de datos y el código de entrenamiento. **Tabla 4-3** enumera las carpetas que se crearán. En el ejemplo, el nombre del bucket y los nombres de las carpetas junto con los nombres reales.

Para obtener detalles sobre cómo crear un bucket de OBS y una carpeta, consulte **Creación de un bucket** y **Creación de una carpeta**.

Asegúrese de que el directorio de OBS que utiliza y ModelArts están en la misma región.

Tabla 4-3 Carpeta para crear

Nombre	Descripción
obs://test-modelarts/pytorch/demo-code/	Almacena el script de entrenamiento.
obs://test-modelarts/pytorch/log/	Almacena los archivos de log de entrenamiento.

Paso 2 Preparar el script de entrenamiento y cargarlo en OBS

Obtenga scripts de entrenamiento **pytorch_synthetic_benchmark.py** y **run_mpi.sh** y cárguelos a **obs://test-modelarts/horovod/demo-code/** en el bucket de OBS.

pytorch_synthetic_benchmark.py es el siguiente:

```
import argparse
import torch.backends.cudnn as cudnn
import torch.nn.functional as F
import torch.optim as optim
import torch.utils.data.distributed
from torchvision import models
import horovod.torch as hvd
import timeit
import numpy as np

# Benchmark settings
parser = argparse.ArgumentParser(description='PyTorch Synthetic Benchmark',

formatter_class=argparse.ArgumentDefaultsHelpFormatter)
parser.add_argument('--fp16-allreduce', action='store_true', default=False,
                    help='use fp16 compression during allreduce')

parser.add_argument('--model', type=str, default='resnet50',
                    help='model to benchmark')
parser.add_argument('--batch-size', type=int, default=32,
                    help='input batch size')

parser.add_argument('--num-warmup-batches', type=int, default=10,
                    help='number of warm-up batches that don\'t count towards
benchmark')
parser.add_argument('--num-batches-per-iter', type=int, default=10,
                    help='number of batches per benchmark iteration')
parser.add_argument('--num-iters', type=int, default=10,
                    help='number of benchmark iterations')

parser.add_argument('--no-cuda', action='store_true', default=False,
                    help='disables CUDA training')

parser.add_argument('--use-adasum', action='store_true', default=False,
                    help='use adasum algorithm to do reduction')

args = parser.parse_args()
args.cuda = not args.no_cuda and torch.cuda.is_available()

hvd.init()

if args.cuda:
    # Horovod: pin GPU to local rank.
    torch.cuda.set_device(hvd.local_rank())

cudnn.benchmark = True
```



```
# Set up standard model.
model = getattr(models, args.model)()

# By default, Adasum doesn't need scaling up learning rate.
lr_scaler = hvd.size() if not args.use_adasum else 1

if args.cuda:
    # Move model to GPU.
    model.cuda()
    # If using GPU Adasum allreduce, scale learning rate by local_size.
    if args.use_adasum and hvd.nccl_built():
        lr_scaler = hvd.local_size()

optimizer = optim.SGD(model.parameters(), lr=0.01 * lr_scaler)

# Horovod: (optional) compression algorithm.
compression = hvd.Compression.fp16 if args.fp16_allreduce else
hvd.Compression.none

# Horovod: wrap optimizer with DistributedOptimizer.
optimizer = hvd.DistributedOptimizer(optimizer,
                                     named_parameters=model.named_parameters(),
                                     compression=compression,
                                     op=hvd.Adasum if args.use_adasum else
hvd.Average)

# Horovod: broadcast parameters & optimizer state.
hvd.broadcast_parameters(model.state_dict(), root_rank=0)
hvd.broadcast_optimizer_state(optimizer, root_rank=0)

# Set up fixed fake data
data = torch.randn(args.batch_size, 3, 224, 224)
target = torch.LongTensor(args.batch_size).random_() % 1000
if args.cuda:
    data, target = data.cuda(), target.cuda()


def benchmark_step():
    optimizer.zero_grad()
    output = model(data)
    loss = F.cross_entropy(output, target)
    loss.backward()
    optimizer.step()


def log(s, nl=True):
    if hvd.rank() != 0:
        return
    print(s, end='\n' if nl else '')


log('Model: %s' % args.model)
log('Batch size: %d' % args.batch_size)
device = 'GPU' if args.cuda else 'CPU'
log('Number of %ss: %d' % (device, hvd.size()))

# Warm-up
log('Running warmup...')
timeit.timeit(benchmark_step, number=args.num_warmup_batches)

# Benchmark
log('Running benchmark...')
img_secs = []
for x in range(args.num_iters):
    time = timeit.timeit(benchmark_step, number=args.num_batches_per_iter)
    img_sec = args.batch_size * args.num_batches_per_iter / time
    log('Iter #%d: %.1f img/sec per %s' % (x, img_sec, device))
    img_secs.append(img_sec)
```



```
# Results
img_sec_mean = np.mean(img_secs)
img_sec_conf = 1.96 * np.std(img_secs)
log('Img/sec per %s: %.1f +-%.1f' % (device, img_sec_mean, img_sec_conf))
log('Total img/sec on %d %s(s): %.1f +-%.1f' %
    (hvd.size(), device, hvd.size() * img_sec_mean, hvd.size() * img_sec_conf))
```

run_mpi.sh es el siguiente:

```
#!/bin/bash
MY_HOME=/home/ma-user

MY_SSHD_PORT=${MY_SSHD_PORT:-"36666"}

MY_MPI_BTL_TCP_IF=${MY_MPI_BTL_TCP_IF:-"eth0,bond0"}

MY_TASK_INDEX=${MA_TASK_INDEX:-${VC_TASK_INDEX:-${VK_TASK_INDEX}}}

MY_MPI_SLOTS=${MY_MPI_SLOTS:-"${MA_NUM_GPUS}"}

MY_MPI_TUNE_FILE="${MY_HOME}/env_for_user_process"

if [ -z ${MY_MPI_SLOTS} ]; then
    echo "[run_mpi] MY_MPI_SLOTS is empty, set it be 1"
    MY_MPI_SLOTS="1"
fi

printf "MY_HOME: ${MY_HOME}\nMY_SSHD_PORT: ${MY_SSHD_PORT}\nMY_MPI_BTL_TCP_IF: $
{MY_MPI_BTL_TCP_IF}\nMY_TASK_INDEX: ${MY_TASK_INDEX}\nMY_MPI_SLOTS: $
{MY_MPI_SLOTS}\n"

env | grep -E '^MA_|SHARED_|^S3_|^PATH|^VC_WORKER_|^SCC|^CRED' | grep -v '=$' > $
{MY_MPI_TUNE_FILE}
# add -x to each line
sed -i 's/^/-x /' ${MY_MPI_TUNE_FILE}

sed -i "s|{{MY_SSHD_PORT}}|${MY_SSHD_PORT}|g" ${MY_HOME}/etc/ssh/sshd_config

# start sshd service
bash -c "$(which sshd) -f ${MY_HOME}/etc/ssh/sshd_config"

# confirm the sshd is up
netstat -anp | grep LIS | grep ${MY_SSHD_PORT}

if [ $MY_TASK_INDEX -eq 0 ]; then
    # generate the hostfile of mpi
    for ((i=0; i<$MA_NUM_HOSTS; i++))
    do
        eval hostname=${MA_VJ_NAME}-${MA_TASK_NAME}-${i}.${MA_VJ_NAME}
        echo "[run_mpi] hostname: ${hostname}"

        ip=""
        while [ -z "$ip" ]; do
            ip=$(ping -c 1 ${hostname} | grep "PING" | sed -E 's/PING .* .
([0-9.]+). .*/\1/g')
            sleep 1
        done
        echo "[run_mpi] resolved ip: ${ip}"

        # test the sshd is up
        while :
        do
            if [ cat < /dev/null >/dev/tcp/${ip}/${MY_SSHD_PORT} ]; then
                break
            fi
            sleep 1
        done

        echo "[run_mpi] the sshd of ip ${ip} is up"
```



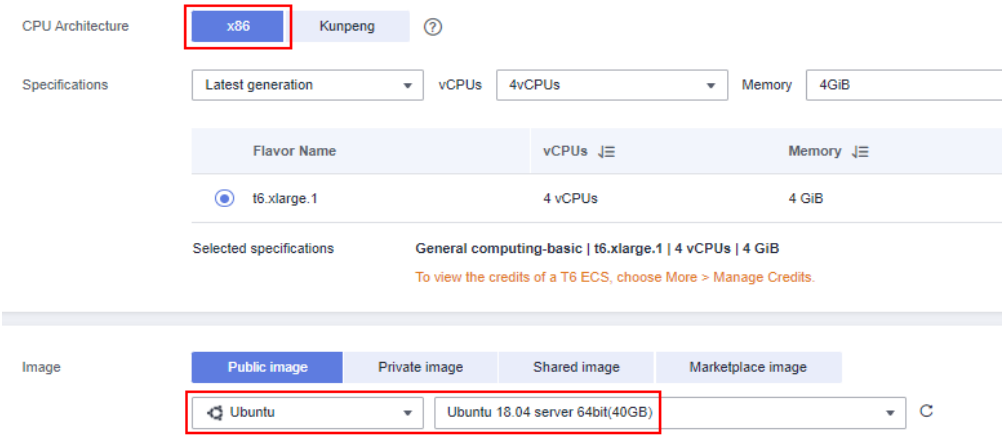
```
        echo "${ip} slots=$MY_MPI_SLOTS" >> ${MY_HOME}/hostfile
    done

    printf "[run_mpi] hostfile:\n`cat ${MY_HOME}/hostfile`\n"
fi

RET_CODE=0

if [ $MY_TASK_INDEX -eq 0 ]; then

    echo "[run_mpi] start exec command time: "$(date +"%Y-%m-%d-%H:%M:%S")

    np=$(( ${MA_NUM_HOSTS} * ${MY_MPI_SLOTS} ))

    echo "[run_mpi] command: mpirun -np ${np} -hostfile ${MY_HOME}/hostfile -mca
plm_rsh_args \"-p ${MY_SSHD_PORT}\" -tune ${MY_MPI_TUNE_FILE} ... $@"

    # execute mpirun at worker-0
    # mpirun
    mpirun \
        -np ${np} \
        -hostfile ${MY_HOME}/hostfile \
        -mca plm_rsh_args "-p ${MY_SSHD_PORT}" \
        -tune ${MY_MPI_TUNE_FILE} \
        -bind-to none -map-by slot \
        -x NCCL_DEBUG=INFO -x NCCL_SOCKET_IFNAME=${MY_MPI_BTL_TCP_IF} -x
NCCL_SOCKET_FAMILY=AF_INET \
        -x HOROVOD_MPI_THREADS_DISABLE=1 \
        -x LD_LIBRARY_PATH \
        -mca pml ob1 -mca btl ^openib -mca plm_rsh_no_tree_spawn true \
        "$@"

    RET_CODE=$?

    if [ $RET_CODE -ne 0 ]; then
        echo "[run_mpi] exec command failed, exited with $RET_CODE"
    else
        echo "[run_mpi] exec command successfully, exited with $RET_CODE"
    fi

    # stop 1...N worker by killing the sleep proc
    sed -i '1d' ${MY_HOME}/hostfile
    if [ `cat ${MY_HOME}/hostfile | wc -l` -ne 0 ]; then
        echo "[run_mpi] stop 1 to (N - 1) worker by killing the sleep proc"

        sed -i 's/${MY_MPI_SLOTS}/1/g' ${MY_HOME}/hostfile
        printf "[run_mpi] hostfile:\n`cat ${MY_HOME}/hostfile`\n"

        mpirun \
        --hostfile ${MY_HOME}/hostfile \
        --mca btl_tcp_if_include ${MY_MPI_BTL_TCP_IF} \
        --mca plm_rsh_args "-p ${MY_SSHD_PORT}" \
        -x PATH -x LD_LIBRARY_PATH \
        pkill sleep \
        > /dev/null 2>&1
    fi

    echo "[run_mpi] exit time: "$(date +"%Y-%m-%d-%H:%M:%S")
else
    echo "[run_mpi] the training log is in worker-0"
    sleep 365d
    echo "[run_mpi] exit time: "$(date +"%Y-%m-%d-%H:%M:%S")
fi

exit $RET_CODE
```

Paso 3 Preparar un servidor

Obtener un servidor Linux x86_64 que ejecute Ubuntu 18.04. Un ECS o su PC local servirán.

Para obtener detalles sobre cómo comprar un ECS, consulte **Compra e inicio de sesión en un ECS de Linux**. Seleccione una imagen pública. Se recomienda una imagen de Ubuntu 18.04.

Figura 4-23 Creación de un ECS con una imagen pública (x86)



Paso 4 Crear una imagen personalizada

Cree una imagen de contenedor con las siguientes configuraciones y utilice la imagen para crear un trabajo de entrenamiento en ModelArts:

- ubuntu-18.04
- cuda-11.1
- python-3.7.13
- mlnx ofed-5.4
- pytorch-1.8.1
- horovod-0.22.1

A continuación se describe cómo crear una imagen personalizada escribiendo un Dockerfile.

1. Instale Docker.

   A continuación se utiliza Linux x86_64 OS como ejemplo para describir cómo obtener el paquete de instalación de Docker. Para obtener detalles sobre cómo instalar Docker, consulte los **documentos oficiales de Docker**. Ejecute los siguientes comandos para instalar Docker:

   ```
   curl -fsSL get.docker.com -o get-docker.sh
   sh get-docker.sh
   ```

   Si se ejecuta el comando **docker images**, Docker se ha instalado. Si es así, omita este paso.

2. Verifique la versión del motor de Docker. Ejecute el siguiente comando:

   ```
   docker version | grep -A 1 Engine
   ```

   Se muestra la siguiente información:

   ```
   Engine:
    Version:          18.09.0
   ```

   NOTA

   Utilice el motor de Docker de la versión anterior o posterior para crear una imagen personalizada.

3. Cree una carpeta denominada **context**.


```
mkdir -p context
```

4. Obtenga el archivo **pip.conf**. En este ejemplo, se utiliza el origen pip proporcionado por Huawei Mirrors, que es el siguiente:

```
[global]
index-url = https://repo.huaweicloud.com/repository/pypi/simple
trusted-host = repo.huaweicloud.com
timeout = 120
```

> NOTA
>
> Para obtener **pip.conf**, acceda a Huawei Mirrors en **https://mirrors.huaweicloud.com/home** y busque **pypi**.

5. Descargue el archivo de código de Horovod de fuente.

   Descargue **horovod-0.22.1.tar.gz** desde https://pypi.org/project/horovod/0.22.1/#files.

6. Descargue los archivos torch*.whl.

   Descargue los siguientes archivos .whl desde https://download.pytorch.org/whl/torch_stable.html:

   – torch-1.8.1+cu111-cp37-cp37m-linux_x86_64.whl
   – torchaudio-0.8.1-cp37-cp37m-linux_x86_64.whl
   – torchvision-0.9.1+cu111-cp37-cp37m-linux_x86_64.whl

   > NOTA
   >
   > El código URL del signo más (+) es %2B. Cuando busque archivos en los sitios web anteriores, sustituya el signo más (+) del nombre del archivo por %2B. Por ejemplo, **torch-1.8.1%2Bcu111-cp37-cp37m-linux_x86_64.whl**.

7. Descargue el archivo de instalación de Miniconda3.

   Descargue el archivo de instalación de Miniconda3 py37 4.12.0 (Python 3.7.13) desde https://repo.anaconda.com/miniconda/Miniconda3-py37_4.12.0-Linux-x86_64.sh.

8. Escribe la imagen del contenedor Dockerfile.

   Cree un archivo vacío denominado **Dockerfile** en la carpeta **context** y copie el siguiente contenido en el archivo:

```
# The server on which the container image is created must access the Internet.

# Base container image at https://github.com/NVIDIA/nvidia-docker/wiki/CUDA
#
# https://docs.docker.com/develop/develop-images/multistage-build/#use-multi-
stage-builds
# require Docker Engine >= 17.05
#
# builder stage
FROM nvidia/cuda:11.1.1-devel-ubuntu18.04 AS builder

# Install CMake obtained from Huawei Mirrors.
RUN cp -a /etc/apt/sources.list /etc/apt/sources.list.bak && \
    sed -i "s@http://.*archive.ubuntu.com@http://
repo.huaweicloud.com@g" /etc/apt/sources.list && \
    sed -i "s@http://.*security.ubuntu.com@http://
repo.huaweicloud.com@g" /etc/apt/sources.list && \
    echo > /etc/apt/apt.conf.d/00skip-verify-peer.conf "Acquire
{ https::Verify-Peer false }" && \
    apt-get update && \
    apt-get install -y build-essential cmake g++-7 && \
    apt-get clean && \
    mv /etc/apt/sources.list.bak /etc/apt/sources.list && \
    rm /etc/apt/apt.conf.d/00skip-verify-peer.conf

# The default user of the base container image is root.
# USER root
```



```
# Use the PyPI configuration obtained from Huawei Mirrors.
RUN mkdir -p /root/.pip/
COPY pip.conf /root/.pip/pip.conf

# Copy the installation files to the /tmp directory in the base container
image.
COPY Miniconda3-py37_4.12.0-Linux-x86_64.sh /tmp
COPY torch-1.8.1+cu111-cp37-cp37m-linux_x86_64.whl /tmp
COPY torchvision-0.9.1+cu111-cp37-cp37m-linux_x86_64.whl /tmp
COPY torchaudio-0.8.1-cp37-cp37m-linux_x86_64.whl /tmp
COPY openmpi-3.0.0-bin.tar.gz /tmp
COPY horovod-0.22.1.tar.gz /tmp

# https://conda.io/projects/conda/en/latest/user-guide/install/
linux.html#installing-on-linux
# Install Miniconda3 in the /home/ma-user/miniconda3 directory of the base
container image.
RUN bash /tmp/Miniconda3-py37_4.12.0-Linux-x86_64.sh -b -p /home/ma-user/
miniconda3

# Install the Open MPI 3.0.0 file obtained from Horovod v0.22.1.
# https://github.com/horovod/horovod/blob/v0.22.1/docker/horovod/Dockerfile
# https://github.com/horovod/horovod/files/1596799/openmpi-3.0.0-bin.tar.gz
RUN cd /usr/local && \
    tar -zxf /tmp/openmpi-3.0.0-bin.tar.gz && \
    ldconfig && \
    mpirun --version

# Environment variables required for building Horovod with PyTorch
ENV HOROVOD_NCCL_INCLUDE=/usr/include \
    HOROVOD_NCCL_LIB=/usr/lib/x86_64-linux-gnu \
    HOROVOD_MPICXX_SHOW="/usr/local/openmpi/bin/mpicxx -show" \
    HOROVOD_GPU_OPERATIONS=NCCL \
    HOROVOD_WITH_PYTORCH=1

# Install the .whl files using default Miniconda3 Python environment /home/ma-
user/miniconda3/bin/pip.
RUN cd /tmp && \
    /home/ma-user/miniconda3/bin/pip install --no-cache-dir \
    /tmp/torch-1.8.1+cu111-cp37-cp37m-linux_x86_64.whl \
    /tmp/torchvision-0.9.1+cu111-cp37-cp37m-linux_x86_64.whl \
    /tmp/torchaudio-0.8.1-cp37-cp37m-linux_x86_64.whl

# Build and install horovod-0.22.1.tar.gz using default Miniconda3 Python
environment /home/ma-user/miniconda3/bin/pip.
RUN cd /tmp && \
    /home/ma-user/miniconda3/bin/pip install --no-cache-dir \
    /tmp/horovod-0.22.1.tar.gz

# Create the container image.
FROM nvidia/cuda:11.1.1-runtime-ubuntu18.04

COPY MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64.tgz /tmp

# Install the vim, cURL, net-tools, MLNX_OFED, and SSH tools obtained from
Huawei Mirrors.
RUN cp -a /etc/apt/sources.list /etc/apt/sources.list.bak && \
    sed -i "s@http://.*archive.ubuntu.com@http://
repo.huaweicloud.com@g" /etc/apt/sources.list && \
    sed -i "s@http://.*security.ubuntu.com@http://
repo.huaweicloud.com@g" /etc/apt/sources.list && \
    echo > /etc/apt/apt.conf.d/00skip-verify-peer.conf "Acquire
{ https::Verify-Peer false }" && \
    apt-get update && \
    apt-get install -y vim curl net-tools iputils-ping libfile-find-rule-perl-
perl \
    openssh-client openssh-server && \
    ssh -V && \
```



```
    mkdir -p /run/sshd && \
    # mlnx ofed
    apt-get install -y python libfuse2 dpatch libnl-3-dev autoconf libnl-
route-3-dev pciutils libnuma1 libpci3 m4 libelf1 debhelper automake graphviz
bison lsof kmod libusb-1.0-0 swig libmnl0 autotools-dev flex chrpath libltdl-
dev && \
    cd /tmp && \
    tar -xvf MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64.tgz && \
    MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64/mlnxofedinstall --user-
space-only --basic --without-fw-update -q && \
    cd - && \
    rm -rf /tmp/* && \
    apt-get clean && \
    mv /etc/apt/sources.list.bak /etc/apt/sources.list && \
    rm /etc/apt/apt.conf.d/00skip-verify-peer.conf

# Install the Open MPI 3.0.0 file obtained from Horovod v0.22.1.
# https://github.com/horovod/horovod/blob/v0.22.1/docker/horovod/Dockerfile
# https://github.com/horovod/horovod/files/1596799/openmpi-3.0.0-bin.tar.gz
COPY openmpi-3.0.0-bin.tar.gz /tmp
RUN cd /usr/local && \
    tar -zxf /tmp/openmpi-3.0.0-bin.tar.gz && \
    ldconfig && \
    mpirun --version

# Add user ma-user (UID = 1000, GID = 100).
# A user group whose GID is 100 exists in the basic container image. User ma-
user can directly run the following command:
RUN useradd -m -d /home/ma-user -s /bin/bash -g 100 -u 1000 ma-user

# Copy the /home/ma-user/miniconda3 directory from the builder stage to the
directory with the same name in the current container image.
COPY --chown=ma-user:100 --from=builder /home/ma-user/miniconda3 /home/ma-
user/miniconda3

# Configure the default user and working directory of the container image.
USER ma-user
WORKDIR /home/ma-user

# Configure sshd to support SSH password-free login.
RUN MA_HOME=/home/ma-user && \
    # setup sshd dir
    mkdir -p ${MA_HOME}/etc && \
    ssh-keygen -f ${MA_HOME}/etc/ssh_host_rsa_key -N '' -t rsa  && \
    mkdir -p ${MA_HOME}/etc/ssh ${MA_HOME}/var/run  && \
    # setup sshd config (listen at {{MY_SSHD_PORT}} port)
    echo "Port {{MY_SSHD_PORT}}\n\
HostKey ${MA_HOME}/etc/ssh_host_rsa_key\n\
AuthorizedKeysFile ${MA_HOME}/.ssh/authorized_keys\n\
PidFile ${MA_HOME}/var/run/sshd.pid\n\
StrictModes no\n\
UsePAM no" > ${MA_HOME}/etc/ssh/sshd_config && \
    # generate ssh key
    ssh-keygen -t rsa -f ${MA_HOME}/.ssh/id_rsa -P '' && \
    cat ${MA_HOME}/.ssh/id_rsa.pub >> ${MA_HOME}/.ssh/authorized_keys && \
    # disable ssh host key checking for all hosts
    echo "Host *\n\
  StrictHostKeyChecking no" > ${MA_HOME}/.ssh/config

# Configure the preset environment variables of the container image.
# Set PYTHONUNBUFFERED to 1 to prevent log loss.
ENV PATH=/home/ma-user/miniconda3/bin:$PATH \
    PYTHONUNBUFFERED=1
```

Para obtener detalles sobre cómo escribir un Dockerfile, consulte los **documentos oficiales de Docker**.

9. Descargue el paquete de instalación de MLNX_OFED.

Vaya a **Linux Drivers**. En la ficha **Download**, seleccione los paquetes de instalación de **Current Versions** y **Archive Versions**. En este ejemplo, seleccione **Archive Versions**, configure **Version** en **5.4-3.5.8.0-LTS**, **OS Distribution** en **Ubuntu**, **OS Distribution Version** en **Ubuntu 18.04**, **Architecture** en **x86_64** y descargue el paquete de instalación **MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64.tgz**.

10. Descarga de **openmpi-3.0.0-bin.tar.gz**.

    Descargue **openmpi-3.0.0-bin.tar.gz** desde https://github.com/horovod/horovod/files/1596799/openmpi-3.0.0-bin.tar.gz.

11. Almacene el archivo de origen de pip, el archivo torch*.whl y el archivo de instalación de Miniconda3 en la carpeta **context**, que es la siguiente:

```
context
├── Dockerfile
├── MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64.tgz
├── Miniconda3-py37_4.12.0-Linux-x86_64.sh
├── horovod-0.22.1.tar.gz
├── openmpi-3.0.0-bin.tar.gz
├── pip.conf
├── torch-1.8.1+cu111-cp37-cp37m-linux_x86_64.whl
├── torchaudio-0.8.1-cp37-cp37m-linux_x86_64.whl
└── torchvision-0.9.1+cu111-cp37-cp37m-linux_x86_64.whl
```

12. Cree la imagen de contenedor. Ejecute el siguiente comando en el directorio donde se almacena el Dockerfile para crear la imagen de contenedor **horovod-pytorch:0.22.1-1.8.1-ofed-cuda11.1**:

```
docker build . -t horovod-pytorch:0.22.1-1.8.1-ofed-cuda11.1
```

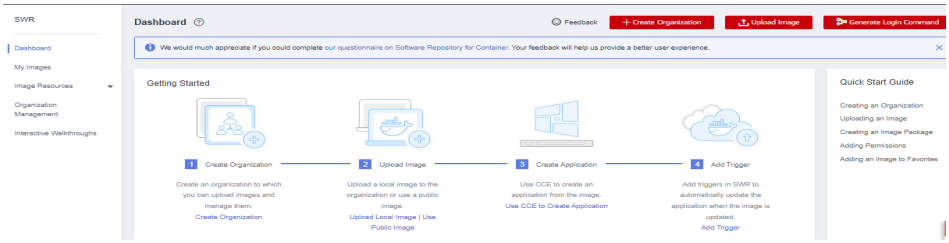
    El siguiente log muestra que se ha creado la imagen.

```
Successfully tagged horovod-pytorch:0.22.1-1.8.1-ofed-cuda11.1
```

Paso 5 Cargar la imagen en SWR

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

Figura 4-24 Consola de SWR

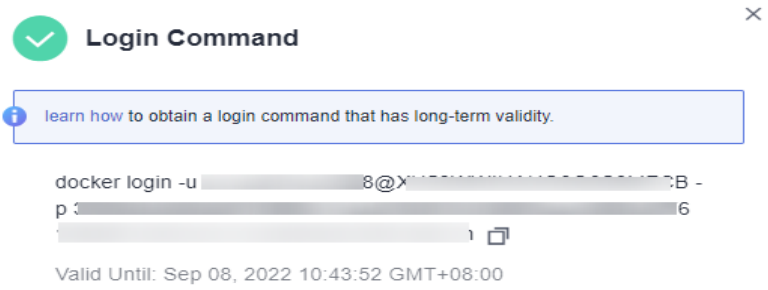


2. Haga clic en **Create Organization** en la esquina superior derecha e introduzca un nombre de organización para crear una organización. Personalice el nombre de la organización. Sustituya el nombre de la organización **deep-learning** en comandos posteriores con el nombre real de la organización.

Figura 4-25 Creación de una organización

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

Figura 4-26 Comando de acceso



4. Inicie sesión en el entorno local como usuario **root** e ingrese el comando de inicio de sesión.
5. Cargue la imagen en SWR.
   a. Etiquete la imagen cargada.
   ```
   # Replace the region, domain, as well as organization name deep-learning with the actual values.
   sudo docker tag horovod-pytorch:0.22.1-1.8.1-ofed-cuda11.1 swr.{region-id}.{domain}/deep-learning/horovod-pytorch:0.22.1-1.8.1-ofed-cuda11.1
   ```
   b. Ejecute el siguiente comando para subir la imagen:
   ```
   # Replace the region, domain, as well as organization name deep-learning with the actual values.
   sudo docker push swr.{region-id}.{domain}/deep-learning/horovod-pytorch:0.22.1-1.8.1-ofed-cuda11.1
   ```
6. 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.

Paso 6 Crear un trabajo de entrenamiento en ModelArts

1. Inicie sesión en la consola de gestión de ModelArts y compruebe si se ha configurado la autorización de acceso para su cuenta. Para obtener más detalles, consulte **Configuración de autorización de delegación**. Si se le ha autorizado mediante claves de acceso, borre la autorización y configure la autorización de delegación.
2. En el panel de navegación, seleccione **Training Management** > **Training Jobs**. La lista de trabajos de entrenamiento se muestra de forma predeterminada.

3. Haga clic en **Create Training Job**. En la página que aparece en pantalla, configure los parámetros y haga clic en **Next**.
   - **Created By**: **Custom algorithms**
   - **Boot Mode**: **Custom images**
   - Ruta de la imagen: imagen creada en **Paso 5 Cargar la imagen en SWR**.
   - **Code Directory**: directorio donde se almacena el archivo de script de arranque en OBS. Por ejemplo, **obs://test-modelarts/pytorch/demo-code/**. El código de entrenamiento se descarga automáticamente en el directorio **${MA_JOB_DIR}/demo-code** del contenedor de entrenamiento. **demo-code** (personalizable) es el directorio de último nivel de la ruta del OBS.
   - **Boot Command**: **bash ${MA_JOB_DIR}/demo-code/run_mpi.sh python ${MA_JOB_DIR}/demo-code/pytorch_synthetic_benchmark.py**. **demo-code** (personalizable) es el directorio de último nivel de la ruta del OBS.
   - **Environment Variable**: haga clic en **Add Environment Variable** y agregue la variable de entorno **MY_SSHD_PORT=38888**.
   - **Resource Pool**: seleccione **Public resource pools**.
   - **Resource Type**: seleccione **GPU**.
   - **Compute Nodes**: 1 o 2
   - **Persistent Log Saving**: habilitado
   - **Job Log Path**: ruta del OBS a los logs de entrenamiento almacenados, por ejemplo, **obs://test-modelarts/pytorch/log/**
4. Confirme las configuraciones del trabajo de entrenamiento y haga clic en **Submit**.
5. Espere hasta que se cree el trabajo de entrenamiento.

   Después de enviar la solicitud de creación de trabajo, el sistema realizará automáticamente operaciones en el backend, como descargar la imagen del contenedor y el directorio de código y ejecutar el comando de arranque. Un trabajo de entrenamiento requiere un cierto período de tiempo para ejecutarse. La duración oscila entre decenas de minutos y varias horas, dependiendo de la lógica del servicio y de los recursos seleccionados. Una vez ejecutado el trabajo de entrenamiento, se muestra un log similar al siguiente.

Figura 4-27 Ejecutar logs de trabajos de entrenamiento con especificaciones de GPU (un nodo de cómputo)

```
58 Iter #0: 342.4 img/sec per GPU
59 Iter #1: 342.7 img/sec per GPU
60 Iter #2: 342.8 img/sec per GPU
61 Iter #3: 342.5 img/sec per GPU
62 Iter #4: 342.9 img/sec per GPU
63 Iter #5: 342.8 img/sec per GPU
64 Iter #6: 342.9 img/sec per GPU
65 Iter #7: 343.0 img/sec per GPU
66 Iter #8: 342.6 img/sec per GPU
67 Iter #9: 342.7 img/sec per GPU
68 Img/sec per GPU: 342.7 +-0.3
69 Total img/sec on 1 GPU(s): 342.7 +-0.3
```

Figura 4-28 Ejecutar logs de trabajos de entrenamiento con especificaciones de GPU (dos nodos de cómputo)

```
84 Iter #0: 115.1 img/sec per GPU
85 Iter #1: 123.5 img/sec per GPU
86 Iter #2: 115.7 img/sec per GPU
87 Iter #3: 117.4 img/sec per GPU
88 Iter #4: 120.6 img/sec per GPU
89 Iter #5: 126.9 img/sec per GPU
90 Iter #6: 119.4 img/sec per GPU
91 Iter #7: 118.4 img/sec per GPU
92 Iter #8: 122.0 img/sec per GPU
93 Iter #9: 118.2 img/sec per GPU
94 Img/sec per GPU: 119.7 +-6.8
95 Total img/sec on 2 GPU(s): 239.4 +-13.5
```

4.5 Ejemplo: creación de una imagen personalizada para entrenamiento (MindSpore y GPU)

En esta sección se describe cómo crear una imagen y utilizarla para los entrenamientos en ModelArts. El motor de IA utilizado en la imagen es MindSpore y los recursos utilizados para el entrenamiento son las GPU.

NOTA

Esta sección solo se aplica a los trabajos de entrenamiento de la nueva versión.

Escenario

En este ejemplo, escriba un Dockerfile para crear una imagen personalizada en un servidor Linux x86_64 que ejecute Ubuntu 18.04.

Cree una imagen contenedora con las siguientes configuraciones y utilícela para crear un trabajo de entrenamiento basado en GPU en ModelArts:

- ubuntu-18.04
- cuda-11.1
- python-3.7.13
- mlnx ofed-5.4
- mindspore gpu-1.8.1

Procedimiento

Antes de utilizar una imagen personalizada para crear un trabajo de entrenamiento, debe estar familiarizado con Docker y tener experiencia en desarrollo.

- **Requisitos previos**
- **Paso 1 Crear un bucket de OBS y una carpeta**
- **Paso 2 Crear un conjunto de datos y cargarlo en OBS**
- **Paso 3 Preparar el script de entrenamiento y cargarlo en OBS**
- **Paso 4 Preparar un servidor**
- **Paso 5 Crear una imagen personalizada**
- **Paso 6 Cargar la imagen en SWR**
- **Paso 7 Crear un trabajo de entrenamiento en ModelArts**

Requisitos previos

Ha creado una cuenta en Huawei Cloud. La cuenta no está en mora ni congelada.

Paso 1 Crear un bucket de OBS y una carpeta

Cree un bucket y una carpeta en OBS para almacenar la muestra de conjunto de datos y el código de entrenamiento. **Tabla 4-4** enumera las carpetas que se crearán. En el ejemplo, el nombre del bucket y los nombres de las carpetas junto con los nombres reales.

Para obtener más detalles, consulte **Creación de un bucket** y **Creación de carpeta**.

Asegúrese de que OBS y ModelArts se encuentren en la misma región.

Tabla 4-4 Carpetas de OBS requeridas

Carpeta	Descripción
obs://test-modelarts/mindspore-gpu/resnet/	Almacena el script de entrenamiento.
obs://test-modelarts/mindspore-gpu/cifar-10-batches-bin/	Almacena archivos de conjuntos de datos.

Carpeta	Descripción
obs://test-modelarts/mindspore-gpu/output/	Almacena archivos de salida de entrenamiento.
obs://test-modelarts/mindspore-gpu/log/	Almacena archivos de log de entrenamiento.

Paso 2 Crear un conjunto de datos y cargarlo en OBS

Vaya a http://www.cs.toronto.edu/~kriz/cifar.html, descargue el paquete **CIFAR-10 binary version (suitable for C programs)**, descomprímalo y cargue los datos de decomposición en el directorio **obs://test-modelarts/mindspore-gpu/cifar-10-batches-bin/** del bucket del OBS.

Paso 3 Preparar el script de entrenamiento y cargarlo en OBS

Obtenga el archivo de ResNet y el script **run_mpi.sh** y cárguelos en **obs://test-modelarts/mindspore-gpu/ResNet/** en el bucket de OBS.

Descargue el archivo de ResNet desde **https://gitee.com/mindspore/models/tree/r1.8/official/cv/resnet**.

run_mpi.sh es el siguiente:

```
#!/bin/bash
MY_HOME=/home/ma-user

MY_SSHD_PORT=${MY_SSHD_PORT:-"36666"}

MY_MPI_BTL_TCP_IF=${MY_MPI_BTL_TCP_IF:-"eth0,bond0"}

MY_TASK_INDEX=${MA_TASK_INDEX:-${VC_TASK_INDEX:-${VK_TASK_INDEX}}}

MY_MPI_SLOTS=${MY_MPI_SLOTS:-"${MA_NUM_GPUS}"}

MY_MPI_TUNE_FILE="${MY_HOME}/env_for_user_process"

if [ -z ${MY_MPI_SLOTS} ]; then
    echo "[run_mpi] MY_MPI_SLOTS is empty, set it be 1"
    MY_MPI_SLOTS="1"
fi

printf "MY_HOME: ${MY_HOME}\nMY_SSHD_PORT: ${MY_SSHD_PORT}\nMY_MPI_BTL_TCP_IF: $
{MY_MPI_BTL_TCP_IF}\nMY_TASK_INDEX: ${MY_TASK_INDEX}\nMY_MPI_SLOTS: $
{MY_MPI_SLOTS}\n"

env | grep -E '^MA_|^SHARED_|^S3_|^PATH|^VC_WORKER_|^SCC|^CRED' | grep -v '=$' > $
{MY_MPI_TUNE_FILE}
# add -x to each line
sed -i 's/^/-x /' ${MY_MPI_TUNE_FILE}

sed -i "s|{{MY_SSHD_PORT}}|${MY_SSHD_PORT}|g" ${MY_HOME}/etc/ssh/sshd_config

# start sshd service
bash -c "$(which sshd) -f ${MY_HOME}/etc/ssh/sshd_config"

# confirm the sshd is up
netstat -anp | grep LIS | grep ${MY_SSHD_PORT}

if [ $MY_TASK_INDEX -eq 0 ]; then
    # generate the hostfile of mpi
    for ((i=0; i<$MA_NUM_HOSTS; i++))
```



```
    do
        eval hostname=${MA_VJ_NAME}-${MA_TASK_NAME}-${i}.${MA_VJ_NAME}
        echo "[run_mpi] hostname: ${hostname}"

        ip=""
        while [ -z "$ip" ]; do
            ip=$(ping -c 1 ${hostname} | grep "PING" | sed -E 's/PING .* .
([0-9.]+). .*/\1/g')
            sleep 1
        done
        echo "[run_mpi] resolved ip: ${ip}"

        # test the sshd is up
        while :
        do
            if [ cat < /dev/null >/dev/tcp/${ip}/${MY_SSHD_PORT} ]; then
                break
            fi
            sleep 1
        done

        echo "[run_mpi] the sshd of ip ${ip} is up"

        echo "${ip} slots=$MY_MPI_SLOTS" >> ${MY_HOME}/hostfile
    done

    printf "[run_mpi] hostfile:\n`cat ${MY_HOME}/hostfile`\n"
fi

RET_CODE=0

if [ $MY_TASK_INDEX -eq 0 ]; then

    echo "[run_mpi] start exec command time: "$(date +"%Y-%m-%d-%H:%M:%S")

    np=$(( ${MA_NUM_HOSTS} * ${MY_MPI_SLOTS} ))

    echo "[run_mpi] command: mpirun -np ${np} -hostfile ${MY_HOME}/hostfile -mca
plm_rsh_args \"-p ${MY_SSHD_PORT}\" -tune ${MY_MPI_TUNE_FILE} ... $@"

    # execute mpirun at worker-0
    # mpirun
    mpirun \
        -np ${np} \
        -hostfile ${MY_HOME}/hostfile \
        -mca plm_rsh_args "-p ${MY_SSHD_PORT}" \
        -tune ${MY_MPI_TUNE_FILE} \
        -bind-to none -map-by slot \
        -x NCCL_DEBUG=INFO -x NCCL_SOCKET_IFNAME=${MY_MPI_BTL_TCP_IF} -x
NCCL_SOCKET_FAMILY=AF_INET \
        -x HOROVOD_MPI_THREADS_DISABLE=1 \
        -x LD_LIBRARY_PATH \
        -mca pml ob1 -mca btl ^openib -mca plm_rsh_no_tree_spawn true \
        "$@"

    RET_CODE=$?

    if [ $RET_CODE -ne 0 ]; then
        echo "[run_mpi] exec command failed, exited with $RET_CODE"
    else
        echo "[run_mpi] exec command successfully, exited with $RET_CODE"
    fi

    # stop 1...N worker by killing the sleep proc
    sed -i '1d' ${MY_HOME}/hostfile
    if [ `cat ${MY_HOME}/hostfile | wc -l` -ne 0 ]; then
        echo "[run_mpi] stop 1 to (N - 1) worker by killing the sleep proc"

        sed -i 's/${MY_MPI_SLOTS}/1/g' ${MY_HOME}/hostfile
```



```
        printf "[run_mpi] hostfile:\n`cat ${MY_HOME}/hostfile`\n"

        mpirun \
        --hostfile ${MY_HOME}/hostfile \
        --mca btl_tcp_if_include ${MY_MPI_BTL_TCP_IF} \
        --mca plm_rsh_args "-p ${MY_SSHD_PORT}" \
        -x PATH -x LD_LIBRARY_PATH \
        pkill sleep \
        > /dev/null 2>&1
    fi

    echo "[run_mpi] exit time: "$(date +"%Y-%m-%d-%H:%M:%S")
else
    echo "[run_mpi] the training log is in worker-0"
    sleep 365d
    echo "[run_mpi] exit time: "$(date +"%Y-%m-%d-%H:%M:%S")
fi

exit $RET_CODE
```

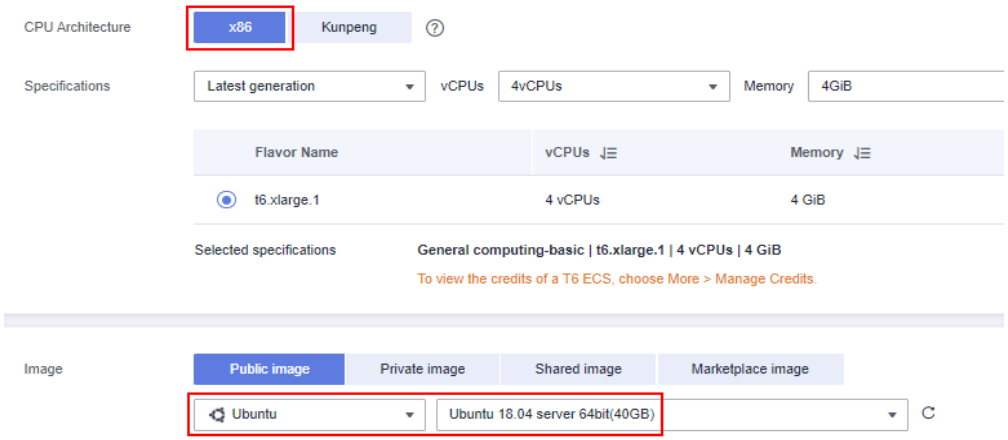
La carpeta **obs://test-modelarts/mindspore-gpu/resnet/** contiene los archivos **resnet** y **run_mpi.sh**.

Paso 4 Preparar un servidor

Obtener un servidor Linux x86_64 que ejecute Ubuntu 18.04. Un ECS o su PC local servirán.

Para obtener detalles sobre cómo comprar un ECS, consulte **Compra e inicio de sesión en un ECS de Linux**. Seleccione una imagen pública. Se recomienda una imagen de Ubuntu 18.04.

Figura 4-29 Creación de un ECS con una imagen pública (x86)



Paso 5 Crear una imagen personalizada

Cree una imagen de contenedor con las siguientes configuraciones y utilice la imagen para crear un trabajo de entrenamiento en ModelArts:

- ubuntu-18.04
- cuda-11.1
- python-3.7.13
- mlnx ofed-5.4
- mindspore gpu-1.8.1

En esta sección se describe cómo escribir un Dockerfile para crear una imagen personalizada.

1. Instale Docker.

   A continuación se utiliza Linux x86_64 como ejemplo para describir cómo obtener un paquete de instalación de Docker. Para obtener más detalles sobre cómo instalar Docker, consulte los **documentos oficiales de Docker**. Ejecute el siguiente comando para instalar Docker:

   ```
   curl -fsSL get.docker.com -o get-docker.sh
   sh get-docker.sh
   ```

   Si se puede ejecutar el comando **docker images**, Docker se ha instalado. Si es así, omita este paso.

2. Verifique la versión del motor de Docker. Ejecute el siguiente comando:

   ```
   docker version | grep -A 1 Engine
   ```

   Se muestra la siguiente información:

   ```
    Engine:
     Version:          18.09.0
   ```

   NOTA

   Utilice el motor de Docker de la versión anterior o posterior para crear una imagen personalizada.

3. Cree una carpeta denominada **context**.

   ```
   mkdir -p context
   ```

4. Obtenga el archivo **pip.conf**. En este ejemplo, se utiliza el origen pip proporcionado por Huawei Mirrors, que es el siguiente:

   ```
   [global]
   index-url = https://repo.huaweicloud.com/repository/pypi/simple
   trusted-host = repo.huaweicloud.com
   timeout = 120
   ```

   NOTA

   Para obtener **pip.conf**, pase a Huawei Mirrors **https://mirrors.huaweicloud.com/home** y busque **pypi**.

5. Descargue **mindspore_gpu-1.8.1-cp37-cp37m-linux_x86_64.whl** desde **https://ms-release.obs.cn-north-4.myhuaweicloud.com/1.8.1/MindSpore/gpu/x86_64/cuda-11.1/mindspore_gpu-1.8.1-cp37-cp37m-linux_x86_64.whl**.

6. Descargue el archivo de instalación de Miniconda3.

   Descargue **Miniconda3-py37_4.12.0-Linux-x86_64.sh** desde https://repo.anaconda.com/miniconda/Miniconda3-py37_4.12.0-Linux-x86_64.sh.

7. Escribe la imagen del contenedor Dockerfile.

   Cree un archivo vacío denominado **Dockerfile** en la carpeta **context** y copie el siguiente contenido en el archivo:

   ```
   # The server on which the container image is created must access the Internet.

   # Base container image at https://github.com/NVIDIA/nvidia-docker/wiki/CUDA
   #
   # https://docs.docker.com/develop/develop-images/multistage-build/#use-multi-stage-builds
   # require Docker Engine >= 17.05
   #
   # builder stage
   FROM nvidia/cuda:11.1.1-devel-ubuntu18.04 AS builder

   # The default user of the base container image is root.
   # USER root

   # Use the PyPI configuration obtained from Huawei Mirrors.
   RUN mkdir -p /root/.pip/
   COPY pip.conf /root/.pip/pip.conf
   ```



```
# Copy the installation files to the /tmp directory in the base container
image.
COPY Miniconda3-py37_4.12.0-Linux-x86_64.sh /tmp
COPY mindspore_gpu-1.8.1-cp37-cp37m-linux_x86_64.whl /tmp

# https://conda.io/projects/conda/en/latest/user-guide/install/
linux.html#installing-on-linux
# Install Miniconda3 in the /home/ma-user/miniconda3 directory of the base
container image.
RUN bash /tmp/Miniconda3-py37_4.12.0-Linux-x86_64.sh -b -p /home/ma-user/
miniconda3

# Install the whl file using default Miniconda3 Python environment /home/ma-
user/miniconda3/bin/pip.
RUN cd /tmp && \
    /home/ma-user/miniconda3/bin/pip install --no-cache-dir \
    /tmp/mindspore_gpu-1.8.1-cp37-cp37m-linux_x86_64.whl \
    easydict PyYAML

# Create the container image.
FROM nvidia/cuda:11.1.1-cudnn8-runtime-ubuntu18.04

COPY MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64.tgz /tmp

# Install the vim, cURL, net-tools, MLNX_OFED, and SSH tools obtained from
Huawei Mirrors.
RUN cp -a /etc/apt/sources.list /etc/apt/sources.list.bak && \
    sed -i "s@http://.*archive.ubuntu.com@http://
repo.huaweicloud.com@g" /etc/apt/sources.list && \
    sed -i "s@http://.*security.ubuntu.com@http://
repo.huaweicloud.com@g" /etc/apt/sources.list && \
    echo > /etc/apt/apt.conf.d/00skip-verify-peer.conf "Acquire
{ https::Verify-Peer false }" && \
    apt-get update && \
    apt-get install -y vim curl net-tools iputils-ping libfile-find-rule-perl-
perl \
    openssh-client openssh-server && \
    ssh -V && \
    mkdir -p /run/sshd && \
    # mlnx ofed
    apt-get install -y python libfuse2 dpatch libnl-3-dev autoconf libnl-
route-3-dev pciutils libnuma1 libpci3 m4 libelf1 debhelper automake graphviz
bison lsof kmod libusb-1.0-0 swig libmnl0 autotools-dev flex chrpath libltdl-
dev && \
    cd /tmp && \
    tar -xvf MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64.tgz && \
    MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64/mlnxofedinstall --user-
space-only --basic --without-fw-update -q && \
    cd - && \
    rm -rf /tmp/* && \
    apt-get clean && \
    mv /etc/apt/sources.list.bak /etc/apt/sources.list && \
    rm /etc/apt/apt.conf.d/00skip-verify-peer.conf

# Install the Open MPI 3.0.0 file obtained from Horovod v0.22.1.
# https://github.com/horovod/horovod/blob/v0.22.1/docker/horovod/Dockerfile
# https://github.com/horovod/horovod/files/1596799/openmpi-3.0.0-bin.tar.gz
COPY openmpi-3.0.0-bin.tar.gz /tmp
RUN cd /usr/local && \
    tar -zxf /tmp/openmpi-3.0.0-bin.tar.gz && \
    ldconfig && \
    mpirun --version

# Add user ma-user (UID = 1000, GID = 100).
# A user group whose GID is 100 exists in the basic container image. User ma-
user can directly run the following command:
RUN useradd -m -d /home/ma-user -s /bin/bash -g 100 -u 1000 ma-user

# Copy the /home/ma-user/miniconda3 directory from the builder stage to the
```



```
directory with the same name in the current container image.
COPY --chown=ma-user:100 --from=builder /home/ma-user/miniconda3 /home/ma-
user/miniconda3

# Configure the default user and working directory of the container image.
USER ma-user
WORKDIR /home/ma-user

# Configure sshd to support SSH password-free login.
RUN MA_HOME=/home/ma-user && \
    # setup sshd dir
    mkdir -p ${MA_HOME}/etc && \
    ssh-keygen -f ${MA_HOME}/etc/ssh_host_rsa_key -N '' -t rsa  && \
    mkdir -p ${MA_HOME}/etc/ssh ${MA_HOME}/var/run  && \
    # setup sshd config (listen at {{MY_SSHD_PORT}} port)
    echo "Port {{MY_SSHD_PORT}}\n\
HostKey ${MA_HOME}/etc/ssh_host_rsa_key\n\
AuthorizedKeysFile ${MA_HOME}/.ssh/authorized_keys\n\
PidFile ${MA_HOME}/var/run/sshd.pid\n\
StrictModes no\n\
UsePAM no" > ${MA_HOME}/etc/ssh/sshd_config && \
    # generate ssh key
    ssh-keygen -t rsa -f ${MA_HOME}/.ssh/id_rsa -P '' && \
    cat ${MA_HOME}/.ssh/id_rsa.pub >> ${MA_HOME}/.ssh/authorized_keys && \
    # disable ssh host key checking for all hosts
    echo "Host *\n\
  StrictHostKeyChecking no" > ${MA_HOME}/.ssh/config

# Configure the preset environment variables of the container image.
# Set PYTHONUNBUFFERED to 1 to prevent log loss.
ENV PATH=/home/ma-user/miniconda3/bin:$PATH \
    LD_LIBRARY_PATH=/usr/local/cuda/lib64:/usr/lib/x86_64-linux-
gnu:$LD_LIBRARY_PATH \
    PYTHONUNBUFFERED=1
```

Para obtener detalles sobre cómo escribir un Dockerfile, consulte los **documentos oficiales de Docker**.

8. Descargue **MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64.tgz**

   Vaya a https://network.nvidia.com/products/infiniband-drivers/linux/mlnx_ofed/ y haga clic en **Download**, configure **Version** en **5.4-3.5.8.0-LTS**, **OSDistributionVersion** en **Ubuntu 18.04** y **Architecture** en **x86_64** y descargue **MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64.tgz**.

9. Descarga de **openmpi-3.0.0-bin.tar.gz**.

   Descargue **openmpi-3.0.0-bin.tar.gz** desde https://github.com/horovod/horovod/files/1596799/openmpi-3.0.0-bin.tar.gz.

10. Almacene el archivo de instalación de Dockerfile y de Miniconda3 en la carpeta **context**, que es la siguiente:

    ```
    context
    ├── Dockerfile
    ├── MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64.tgz
    ├── Miniconda3-py37_4.12.0-Linux-x86_64.sh
    ├── mindspore_gpu-1.8.1-cp37-cp37m-linux_x86_64.whl
    ├── openmpi-3.0.0-bin.tar.gz
    └── pip.conf
    ```

11. Cree la imagen de contenedor. Ejecute el siguiente comando en el directorio donde se almacena Dockerfile para crear la imagen de contenedor **mindspore:1.8.1-ofed-cuda11.1**:

    ```
    docker build . -t mindspore:1.8.1-ofed-cuda11.1
    ```

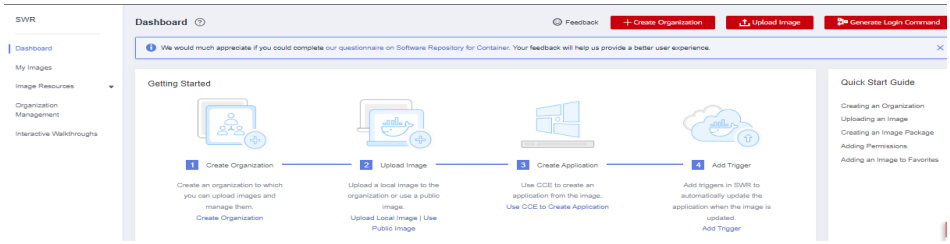
    El siguiente log muestra que se ha creado la imagen.

    ```
    Successfully tagged mindspore:1.8.1-ofed-cuda11.1
    ```


Paso 6 Cargar la imagen en SWR

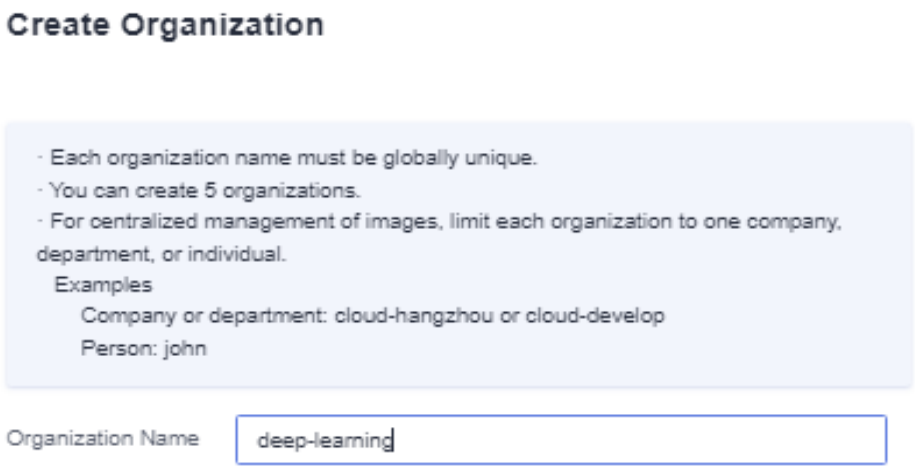
1. Inicie sesión en la consola de SWR y seleccione la región de destino.

Figura 4-30 Consola de SWR



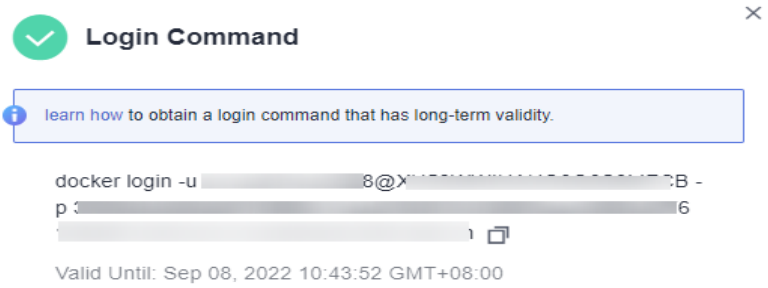
2. Haga clic en **Create Organization** en la esquina superior derecha e introduzca un nombre de organización para crear una organización. Personalice el nombre de la organización. Sustituya el nombre de la organización **deep-learning** en comandos posteriores con el nombre real de la organización.

Figura 4-31 Creación de una organización



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

Figura 4-32 Comando de acceso



4. Inicie sesión en el entorno local como usuario **root** e ingrese el comando de inicio de sesión.
5. Cargue la imagen en SWR.

a. Etiquete la imagen cargada.

```
# Replace the region, domain, as well as organization name deep-learning
with the actual values.
sudo docker tag mindspore:1.8.1-ofed-cuda11.1 swr.{region-id}.{domain}/
deep-learning/mindspore:1.8.1-ofed-cuda11.1
```

b. Ejecute el siguiente comando para subir la imagen:

```
# Replace the region, domain, as well as organization name deep-learning
with the actual values.
sudo docker push swr.{region-id}.{domain}/deep-learning/mindspore:1.8.1-
ofed-cuda11.1
```

6. 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.

Paso 7 Crear un trabajo de entrenamiento en ModelArts

1. Inicie sesión en la consola de gestión de ModelArts y compruebe si se ha configurado la autorización de acceso para su cuenta. Para obtener más detalles, consulte **Configuración de autorización de delegación**. Si se le ha autorizado mediante claves de acceso, borre la autorización y configure la autorización de delegación.
2. En el panel de navegación, seleccione **Training Management** > **Training Jobs**. La lista de trabajos de entrenamiento se muestra de forma predeterminada.
3. Haga clic en **Create Training Job**. En la página que aparece en pantalla, configure los parámetros y haga clic en **Next**.
   - **Created By**: **Custom algorithms**
   - **Boot Mode**: **Custom images**
   - Ruta de la imagen: imagen creada en **Paso 6 Cargar la imagen en SWR**.
   - **Code Directory**: directorio donde se almacena el archivo de script de arranque en OBS, por ejemplo, **obs://test-modelarts/mindspore-gpu/resnet/**. El código de entrenamiento se descarga automáticamente en el directorio **${MA_JOB_DIR}/resnet** del contenedor de entrenamiento. **resnet** (personalizable) es el directorio de último nivel de la ruta de OBS.
   - **Boot Command**: **bash ${MA_JOB_DIR}/resnet/run_mpi.sh python ${MA_JOB_DIR}/resnet/train.py**. **resnet** (personalizable) es el directorio de último nivel de la ruta de OBS.
   - **Training Input**: haga clic en **Add Training Input**. Escriba **data_path** como nombre, seleccione la ruta de OBS al conjunto de datos de destino, por ejemplo, **obs://test-modelarts/mindspore-gpu/cifar-10-batches-bin/** y configure **Obtained from** como **Hyperparameters**.
   - **Training Output**: haga clic en **Add Training Output**. Escriba **output_path** como nombre, seleccione una ruta de OBS para almacenar salidas de entrenamiento, por ejemplo **obs://test-modelarts/mindspore-gpu/output/** y configure **Obtained from** como **Hyperparameters** y **Predownload** como **No**.
   - **Hyperparameters**: haga clic en **Add Hyperparameter** y agregue los siguientes hiperparámetros:
     - run_distribute=True
     - device_num=1 (Set this parameter based on the number of GPUs in the instance flavors.)
     - device_target=GPU
     - epoch_size=2

- **Environment Variable**: haga clic en **Add Environment Variable** y agregue la variable de entorno **MY_SSHD_PORT=38888**.
- **Resource Pool**: seleccione **Public resource pools**.
- **Resource Type**: seleccione **GPU**.
- **Compute Nodes**: 1 o 2
- **Persistent Log Saving**: habilitado
- **Job Log Path**: ruta de OBS a los logs de entrenamiento almacenados, por ejemplo, **obs://test-modelarts/mindspore-gpu/log/**

4. Confirme las configuraciones del trabajo de entrenamiento y haga clic en **Submit**.
5. Espere hasta que se cree el trabajo de entrenamiento.

   Después de enviar la solicitud de creación de trabajo, el sistema realizará automáticamente operaciones en el backend, como descargar la imagen del contenedor y el directorio de código y ejecutar el comando de arranque. Un trabajo de entrenamiento requiere un cierto período de tiempo para ejecutarse. La duración oscila entre decenas de minutos y varias horas, dependiendo de la lógica del servicio y de los recursos seleccionados. Una vez ejecutado el trabajo de entrenamiento, se muestra un log similar al siguiente.

   Figura 4-33 Ejecutar logs de trabajos de entrenamiento con especificaciones de GPU (un nodo de cómputo)

```
127 epoch: 1 step: 1875, loss is 1.4800076
128 Train epoch time: 369422.027 ms, per step time: 197.025 ms
129 epoch: 2 step: 1875, loss is 1.0306032
130 Train epoch time: 66996.087 ms, per step time: 35.731 ms
```

   Figura 4-34 Ejecutar logs de trabajos de entrenamiento con especificaciones de GPU (dos nodos de cómputo)

```
187 epoch: 1 step: 937, loss is 1.8482083
188 epoch: 1 step: 937, loss is 1.342748
189 Train epoch time: 488492.170 ms, per step time: 521.336 ms
190 Train epoch time: 488490.528 ms, per step time: 521.335 ms
191 epoch: 2 step: 937, loss is 0.9150252
192 Train epoch time: 180200.654 ms, per step time: 192.317 ms
193 epoch: 2 step: 937, loss is 0.9933052
194 Train epoch time: 180199.969 ms, per step time: 192.316 ms
195 172.16.0.45 slots=1
```

4.6 Ejemplo: creación de una imagen personalizada para entrenamiento (TensorFlow y GPU)

En esta sección se describe cómo crear una imagen y utilizarla para los entrenamientos en ModelArts. El motor de IA utilizado en la imagen es TensorFlow y los recursos utilizados para el entrenamiento son las GPU.

NOTA

Esta sección solo se aplica a los trabajos de entrenamiento de la nueva versión.

Escenario

En este ejemplo, escriba un Dockerfile para crear una imagen personalizada en un servidor Linux x86_64 que ejecute Ubuntu 18.04.

Cree una imagen contenedora con las siguientes configuraciones y utilícela para crear un trabajo de entrenamiento basado en GPU en ModelArts:

- ubuntu-18.04
- cuda-11.2
- python-3.7.13
- mlnx ofed-5.4
- tensorflow gpu-2.10.0

Procedimiento

Antes de utilizar una imagen personalizada para crear un trabajo de entrenamiento, debe estar familiarizado con Docker y tener experiencia en desarrollo.

1. **Requisitos previos**
2. **Paso 1 Crear un bucket de OBS y una carpeta**
3. **Paso 2 Crear un conjunto de datos y cargarlo en OBS**
4. **Paso 3 Preparar el script de entrenamiento y cargarlo en OBS**
5. **Paso 4 Preparar un servidor**
6. **Paso 5 Crear una imagen personalizada**
7. **Paso 6 Cargar la imagen en SWR**
8. **Paso 7 Crear un trabajo de entrenamiento en ModelArts**

Requisitos previos

Ha creado una cuenta en Huawei Cloud. La cuenta no está en mora ni congelada.

Paso 1 Crear un bucket de OBS y una carpeta

Cree un bucket y una carpeta en OBS para almacenar la muestra de conjunto de datos y el código de entrenamiento. **Tabla 4-5** enumera las carpetas que se crearán. En el ejemplo, el nombre del bucket y los nombres de las carpetas junto con los nombres reales.

Para obtener detalles sobre cómo crear un bucket de OBS y una carpeta, consulte **Creación de un bucket** y **Creación de una carpeta**.

Asegúrese de que el directorio de OBS que utiliza y ModelArts están en la misma región.

Tabla 4-5 Carpetas de OBS requeridas

Carpeta	Descripción
obs://test-modelarts/tensorflow/code/	Almacena el script de entrenamiento.
obs://test-modelarts/tensorflow/data/	Almacena archivos de conjuntos de datos.

Carpeta	Descripción
obs://test-modelarts/tensorflow/log/	Almacena los archivos de log de entrenamiento.

Paso 2 Crear un conjunto de datos y cargarlo en OBS

Descargue **mnist.npz** desde **https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz** y cárguelo en **obs://test-modelarts/tensorflow/data/** en el bucket de OBS.

Paso 3 Preparar el script de entrenamiento y cargarlo en OBS

Obtenga el script de entrenamiento **mnist.py** y cárguelo en **obs://test-modelarts/tensorflow/code/** en el bucket de OBS.

mnist.py es el siguiente:

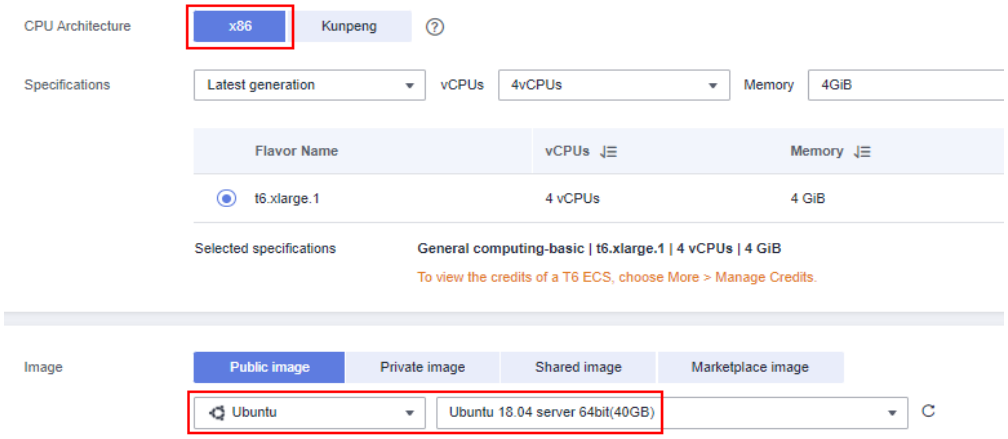
```
import argparse
import tensorflow as tf

parser = argparse.ArgumentParser(description='TensorFlow quick start')
parser.add_argument('--data_url', type=str, default="./Data", help='path where
the dataset is saved')
args = parser.parse_args()

mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data(args.data_url)
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10)
])

loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)

model.compile(optimizer='adam',
              loss=loss_fn,
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
```

Paso 4 Preparar un servidor

Obtener un servidor Linux x86_64 que ejecute Ubuntu 18.04. Un ECS o su PC local servirán.

Para obtener detalles sobre cómo comprar un ECS, consulte **Compra e inicio de sesión en un ECS de Linux**. Seleccione una imagen pública. Se recomienda una imagen de Ubuntu 18.04.

Figura 4-35 Creación de un ECS con una imagen pública (x86)



Paso 5 Crear una imagen personalizada

Cree una imagen de contenedor con las siguientes configuraciones y utilice la imagen para crear un trabajo de entrenamiento en ModelArts:

- ubuntu-18.04
- cuda-11.1
- python-3.7.13
- mlnx ofed-5.4
- mindspore gpu-1.8.1

A continuación se describe cómo crear una imagen personalizada escribiendo un Dockerfile.

1. Instale Docker.

   A continuación se utiliza Linux x86_64 OS como ejemplo para describir cómo obtener el paquete de instalación de Docker. Para obtener detalles sobre cómo instalar Docker, consulte los **documentos oficiales de Docker**. Ejecute los siguientes comandos para instalar Docker:

   ```
   curl -fsSL get.docker.com -o get-docker.sh
   sh get-docker.sh
   ```

   Si se ejecuta el comando **docker images**, Docker se ha instalado. Si es así, omita este paso.

2. Verifique la versión del motor de Docker. Ejecute el siguiente comando:

   ```
   docker version | grep -A 1 Engine
   ```

   Se muestra la siguiente información:

   ```
   Engine:
    Version:          18.09.0
   ```

   NOTA

   Utilice el motor de Docker de la versión anterior o posterior para crear una imagen personalizada.

3. Cree una carpeta denominada **context**.

   ```
   mkdir -p context
   ```

4. Obtenga el archivo **pip.conf**. En este ejemplo, se utiliza el origen pip proporcionado por Huawei Mirrors, que es el siguiente:

   ```
   [global]
   index-url = https://repo.huaweicloud.com/repository/pypi/simple
   ```



```
trusted-host = repo.huaweicloud.com
timeout = 120
```

NOTA

Para obtener **pip.conf**, acceda a Huawei Mirrors en https://mirrors.huaweicloud.com/home y busque **pypi**.

5. Descargue **tensorflow_gpu-2.10.0-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl**.

   Descargue **tensorflow_gpu-2.10.0-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl** desde **https://pypi.org/project/tensorflow-gpu/2.10.0/#files**.

6. Descargue el archivo de instalación de Miniconda3.

   Descargue el archivo de instalación de Miniconda3 py37 4.12.0 (Python 3.7.13) desde https://repo.anaconda.com/miniconda/Miniconda3-py37_4.12.0-Linux-x86_64.sh.

7. Escribe la imagen del contenedor Dockerfile.

   Cree un archivo vacío denominado **Dockerfile** en la carpeta **context** y copie el siguiente contenido en el archivo:

```
# The server on which the container image is created must access the Internet.

# Base container image at https://github.com/NVIDIA/nvidia-docker/wiki/CUDA
#
# https://docs.docker.com/develop/develop-images/multistage-build/#use-multi-stage-builds
# require Docker Engine >= 17.05
#
# builder stage
FROM nvidia/cuda:11.2.2-cudnn8-runtime-ubuntu18.04 AS builder

# The default user of the base container image is root.
# USER root

# Use the PyPI configuration obtained from Huawei Mirrors.
RUN mkdir -p /root/.pip/
COPY pip.conf /root/.pip/pip.conf

# Copy the installation files to the /tmp directory in the base container image.
COPY Miniconda3-py37_4.12.0-Linux-x86_64.sh /tmp
COPY tensorflow_gpu-2.10.0-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl /tmp

# https://conda.io/projects/conda/en/latest/user-guide/install/linux.html#installing-on-linux
# Install Miniconda3 in the /home/ma-user/miniconda3 directory of the base container image.
RUN bash /tmp/Miniconda3-py37_4.12.0-Linux-x86_64.sh -b -p /home/ma-user/miniconda3

# Install the TensorFlow .whl file using default Miniconda3 Python environment /home/ma-user/miniconda3/bin/pip.
RUN cd /tmp && \
    /home/ma-user/miniconda3/bin/pip install --no-cache-dir \
    /tmp/tensorflow_gpu-2.10.0-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl

RUN cd /tmp && \
    /home/ma-user/miniconda3/bin/pip install --no-cache-dir keras==2.10.0

# Create the container image.
FROM nvidia/cuda:11.2.2-cudnn8-runtime-ubuntu18.04

COPY MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64.tgz /tmp
```



```
# Install the vim, cURL, net-tools, and MLNX_OFED tools obtained from Huawei
Mirrors.
RUN cp -a /etc/apt/sources.list /etc/apt/sources.list.bak && \
    sed -i "s@http://.*archive.ubuntu.com@http://
repo.huaweicloud.com@g" /etc/apt/sources.list && \
    sed -i "s@http://.*security.ubuntu.com@http://
repo.huaweicloud.com@g" /etc/apt/sources.list && \
    echo > /etc/apt/apt.conf.d/00skip-verify-peer.conf "Acquire
{ https::Verify-Peer false }" && \
    apt-get update && \
    apt-get install -y vim curl net-tools iputils-ping && \
    # mlnx ofed
    apt-get install -y python libfuse2 dpatch libnl-3-dev autoconf libnl-
route-3-dev pciutils libnuma1 libpci3 m4 libelf1 debhelper automake graphviz
bison lsof kmod libusb-1.0-0 swig libmnl0 autotools-dev flex chrpath libltdl-
dev && \
    cd /tmp && \
    tar -xvf MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64.tgz && \
    MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64/mlnxofedinstall --user-
space-only --basic --without-fw-update -q && \
    cd - && \
    rm -rf /tmp/* && \
    apt-get clean && \
    mv /etc/apt/sources.list.bak /etc/apt/sources.list && \
    rm /etc/apt/apt.conf.d/00skip-verify-peer.conf

# Add user ma-user (UID = 1000, GID = 100).
# A user group whose GID is 100 exists in the base container image. User ma-
user can directly run the following command:
RUN useradd -m -d /home/ma-user -s /bin/bash -g 100 -u 1000 ma-user

# Copy the /home/ma-user/miniconda3 directory from the builder stage to the
directory with the same name in the current container image.
COPY --chown=ma-user:100 --from=builder /home/ma-user/miniconda3 /home/ma-
user/miniconda3

# Configure the default user and working directory of the container image.
USER ma-user
WORKDIR /home/ma-user

# Configure the preset environment variables of the container image.
# Set PYTHONUNBUFFERED to 1 to prevent log loss.
ENV PATH=/home/ma-user/miniconda3/bin:$PATH \
    LD_LIBRARY_PATH=/usr/local/cuda/lib64:/usr/lib/x86_64-linux-
gnu:$LD_LIBRARY_PATH \
    PYTHONUNBUFFERED=1
```

Para obtener detalles sobre cómo escribir un Dockerfile, consulte los **documentos oficiales de Docker**.

8. Descargue **MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64.tgz**.

   Vaya a **Linux Drivers**. En la ficha **Download**, configure **Version** en **5.4-3.5.8.0-LTS**, **OS Distribution Version** en **Ubuntu 18.04**, **Architecture** en **x86_64** y descargue **MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64.tgz**.

9. Almacene el archivo de instalación de Dockerfile y de Miniconda3 en la carpeta **context**, que es la siguiente:

```
context
├── Dockerfile
├── MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64.tgz
├── Miniconda3-py37_4.12.0-Linux-x86_64.sh
├── pip.conf
└── tensorflow_gpu-2.10.0-cp37-cp37m-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl
```

10. Cree la imagen de contenedor. Ejecute el siguiente comando en el directorio donde se almacena el Dockerfile para crear la imagen de contenedor **tensorflow:2.10.0-ofed-cuda11.2**:


```
docker build . -t tensorflow:2.10.0-ofed-cuda11.2
```

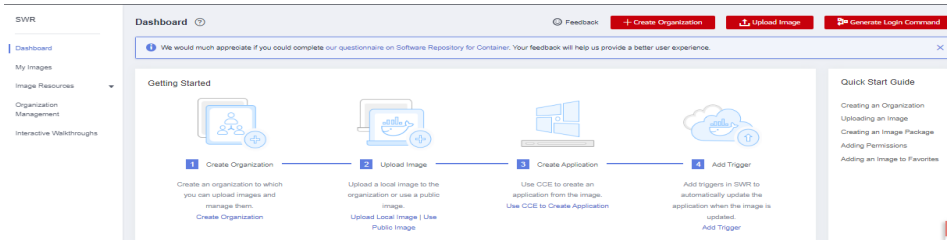
El siguiente log muestra que se ha creado la imagen.

```
Successfully tagged tensorflow:2.10.0-ofed-cuda11.2
```

Paso 6 Cargar la imagen en SWR

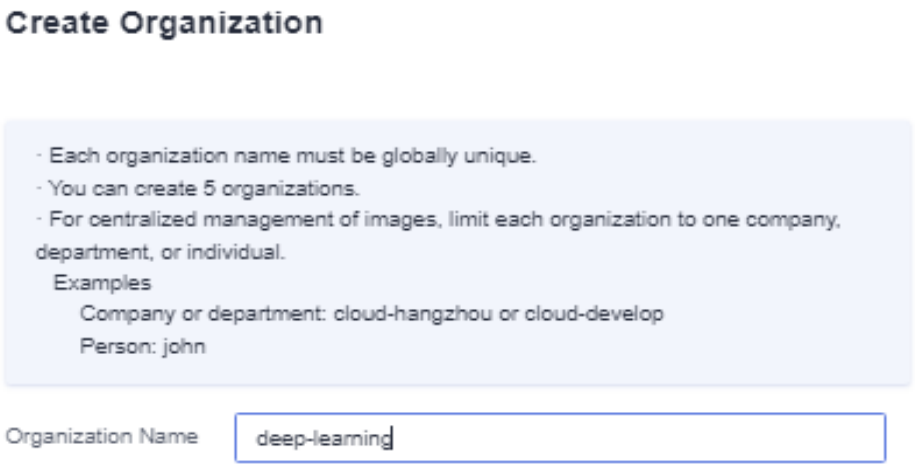
1. Inicie sesión en la consola de SWR y seleccione la región de destino.

Figura 4-36 Consola de SWR



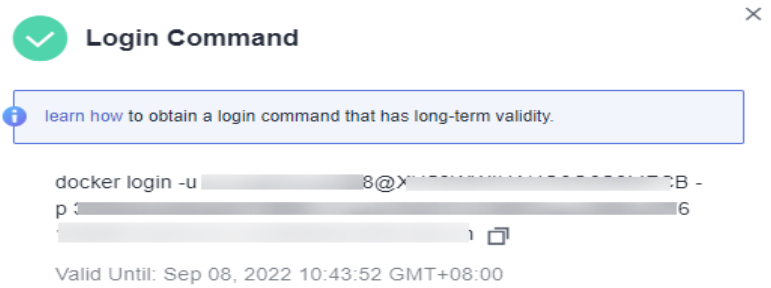
2. Haga clic en **Create Organization** en la esquina superior derecha e introduzca un nombre de organización para crear una organización. Personalice el nombre de la organización. Sustituya el nombre de la organización **deep-learning** en comandos posteriores con el nombre real de la organización.

Figura 4-37 Creación de una organización



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

Figura 4-38 Comando de acceso



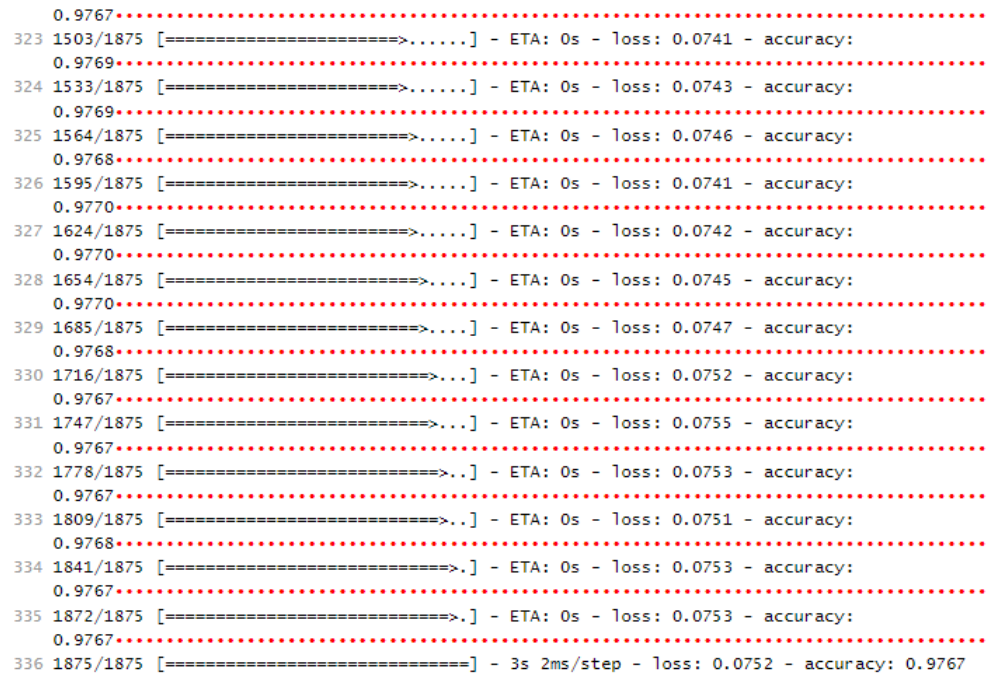
4. Inicie sesión en el entorno local como usuario **root** e ingrese el comando de inicio de sesión.
5. Cargue la imagen en SWR.
   a. Etiquete la imagen cargada.
   ```
   # Replace the region, domain, as well as organization name deep-learning
   with the actual values.
   sudo docker tag tensorflow:2.10.0-ofed-cuda11.2 swr.{region-id}.{domain}/
   deep-learning/tensorflow:2.10.0-ofed-cuda11.2
   ```
   b. Ejecute el siguiente comando para subir la imagen:
   ```
   # Replace the region, domain, as well as organization name deep-learning
   with the actual values.
   sudo docker push swr.{region-id}.{domain}/deep-learning/
   tensorflow:2.10.0-ofed-cuda11.2
   ```
6. 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.

Paso 7 Crear un trabajo de entrenamiento en ModelArts

1. Inicie sesión en la consola de gestión de ModelArts y compruebe si se ha configurado la autorización de acceso para su cuenta. Para obtener más detalles, consulte **Configuración de autorización de delegación**. Si se le ha autorizado mediante claves de acceso, borre la autorización y configure la autorización de delegación.
2. En el panel de navegación, seleccione **Training Management** > **Training Jobs**. La lista de trabajos de entrenamiento se muestra de forma predeterminada.
3. Haga clic en **Create Training Job**. En la página que aparece en pantalla, configure los parámetros y haga clic en **Next**.
   - **Created By**: **Custom algorithms**
   - **Boot Mode**: **Custom images**
   - Ruta de la imagen: imagen creada en **Paso 5 Crear una imagen personalizada**.
   - **Code Directory**: directorio donde se almacena el archivo de script de inicio en OBS, por ejemplo, **obs://test-modelarts/tensorflow/code/**. El código de entrenamiento se descarga automáticamente en el directorio **${MA_JOB_DIR}/code** del contenedor de entrenamiento. **code** (personalizable) es el directorio de último nivel de la ruta del OBS.
   - **Boot Command**: **python ${MA_JOB_DIR}/code/mnist.py**. **code** (personalizable) es el directorio de último nivel de la ruta del OBS.
   - **Training Input**: haga clic en **Add Training Input**. Escriba **data_path** como nombre, seleccione la ruta del OBS a **mnist.npz**. Por ejemplo, **obs://test-modelarts/tensorflow/data/mnist.npz** y configure **Obtained from** como **Hyperparameters**.
   - **Resource Pool**: seleccione **Public resource pools**.
   - **Resource Type**: seleccione **GPU**.
   - **Compute Nodes**: Ingrese **1**.
   - **Persistent Log Saving**: habilitado
   - **Job Log Path**: ruta de OBS a los logs de entrenamiento almacenados, por ejemplo, **obs://test-modelarts/mindspore-gpu/log/**
4. Confirme las configuraciones del trabajo de entrenamiento y haga clic en **Submit**.
5. Espere hasta que se cree el trabajo de entrenamiento.

Después de enviar la solicitud de creación de trabajo, el sistema realizará automáticamente operaciones en el backend, como descargar la imagen del contenedor y el directorio de código y ejecutar el comando de arranque. Un trabajo de entrenamiento requiere un cierto período de tiempo para ejecutarse. La duración oscila entre decenas de minutos y varias horas, dependiendo de la lógica del servicio y de los recursos seleccionados. Una vez ejecutado el trabajo de entrenamiento, se muestra un log similar al siguiente.

Figura 4-39 Ejecutar logs de trabajos de entrenamiento con especificaciones de GPU



5 Inferencia del modelo

5.1 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:

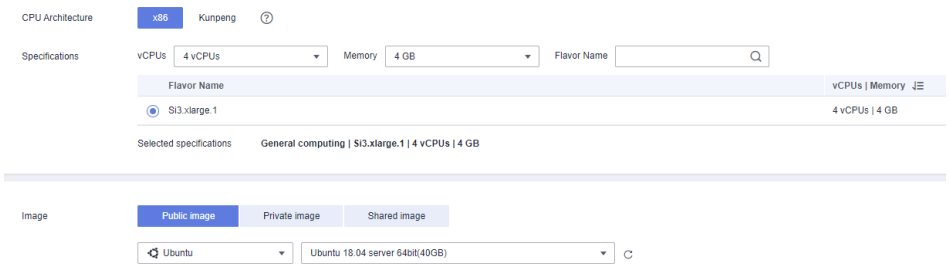
1. **Construcción de una imagen localmente**: Cree un paquete de imágenes personalizadas localmente. Para obtener más información, consulte las **Especificaciones de imágenes personalizadas para crear aplicaciones de AI**.
2. **Verificación de la imagen localmente y su carga a SWR**: Verifique las API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

Para obtener detalles sobre cómo comprar un ECS, consulte **Compra e inicio de sesión en un ECS de Linux**. Al crear el ECS, seleccione una imagen pública de Ubuntu 18.04.

Figura 5-1 Creación de un ECS utilizando una imagen pública de x86



1. Después de iniciar sesión en el host, instale Docker. Para obtener más información, consulte los **documentos oficiales de Docker**. Como alternativa, ejecute los siguientes comandos para instalar Docker:

   ```
   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():
     ```



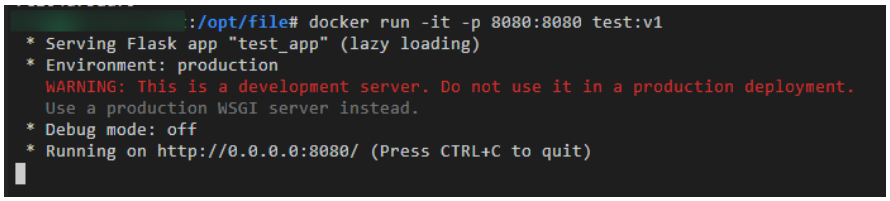
```
    print("----------- in goodbye func ----------")
    return '\nGoodbye!\n'


@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)
```

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 images** 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
   ```

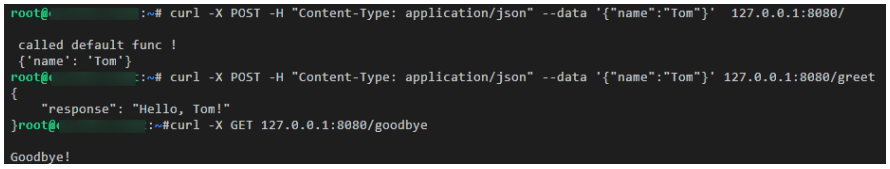
   Figura 5-2 Inicio de una imagen personalizada

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 5-3 Prueba de las funciones de API

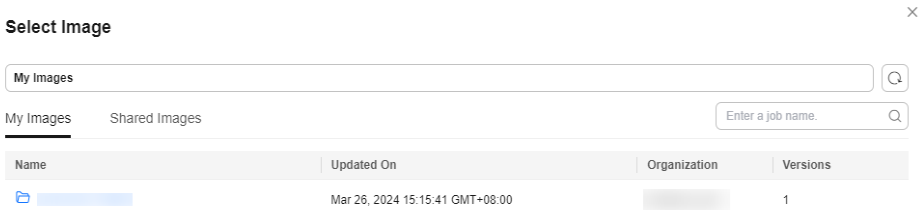
3. Cargue la imagen personalizada en SWR. Para obtener más información, consulte **¿Cómo puedo subir imágenes a SWR?**
4. Vea la imagen cargada en la página **My Images** > **Private Images** de la consola de SWR.

Uso de la imagen personalizada para crear una aplicación de IA

Importe un metamodelo. Para obtener más detalles, consulte **Creación e importación de un Image del modelo**. Los parámetros clave son los siguientes:

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

    Figura 5-4 Imagen privada creada

  - **Container API**: Protocolo y número de puerto para iniciar un modelo. Asegúrese de que el protocolo y el número de puerto son los mismos que los proporcionados en la imagen personalizada.
  - **Image Replication**: indicates whether to copy the model image in the container image to ModelArts. 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 de IA va a fallar.
- **APIs**: API de una imagen personalizada. Este parámetro es opcional. Las API de modelo deben cumplir con las especificaciones de ModelArts. Para obtener más detalles, consulte **Especificaciones para editar un archivo de configuración de modelo**.

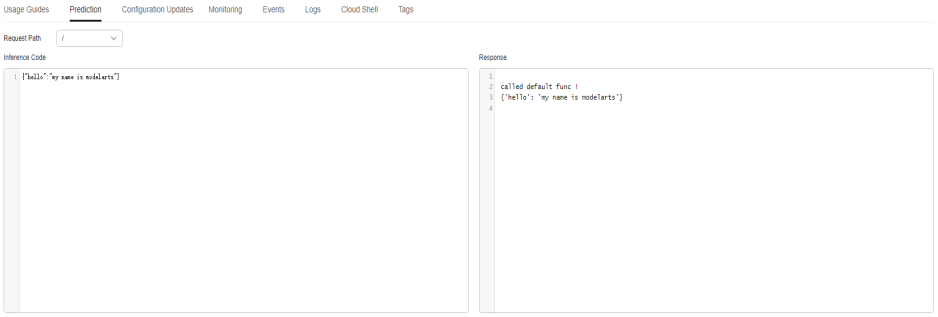
  El archivo de configuración es el siguiente:

```
[{
        "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. Deploy the AI application as a real-time service. Para obtener más detalles, consulte **Despliegue como un servicio en tiempo real**.
2. Vea los detalles sobre el servicio en tiempo real.
3. Acceda al servicio en tiempo real en la página de ficha **Prediction**.

Figura 5-5 Acceso a un servicio en tiempo real



5.2 Habilitación de un servicio de inferencia para acceder a Internet

En esta sección se describe cómo habilitar un servicio de inferencia para acceder a Internet.

Escenarios de aplicación

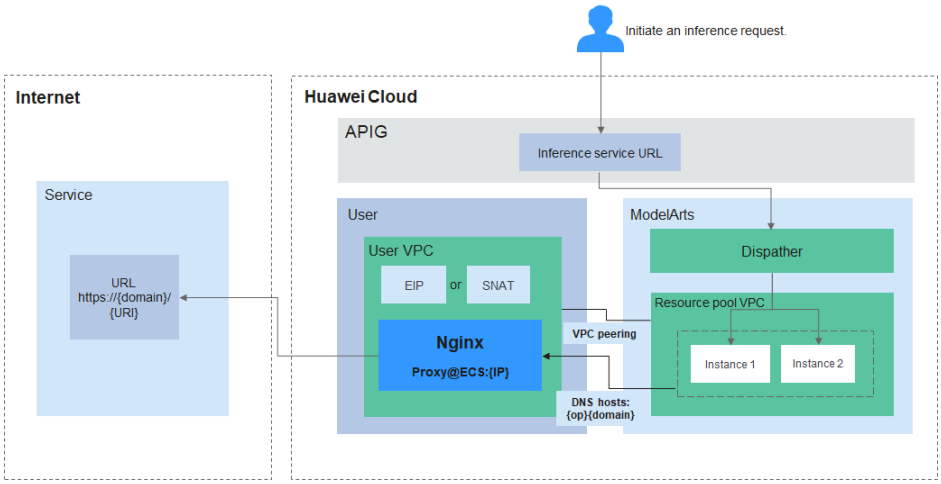
Un servicio de inferencia accede a Internet en los siguientes escenarios:

- Después de ingresar una imagen, el servicio de inferencia llama al OCR en Internet y luego procesa los datos mediante NLP.
- El servicio de inferencia descarga archivos de Internet y los analiza.
- El servicio de inferencia devuelve el resultado del análisis al terminal por Internet.

Diseño de soluciones

Utilice el algoritmo en la instancia en la que se despliega el servicio de inferencia para acceder a Internet.

Figura 5-6 Redes para que un servicio de inferencia acceda a Internet



Procedimiento

- **Configure la red para el grupo de recursos de ModelArts.**
- **Instale y configure un proxy de reenvío para su VPC.**
- **Configure el proxy de DNS y la URL de acceso a Internet en la imagen del algoritmo.**

Paso 1 Configure la red para el grupo de recursos de ModelArts.

Al adquirir un grupo de recursos dedicado, puede seleccionar servicios de inferencia en **Job Type**. En este caso, la red seleccionada debe ser accesible a la VPC de destino.

Figura 5-7 Compra de un grupo de recursos dedicado

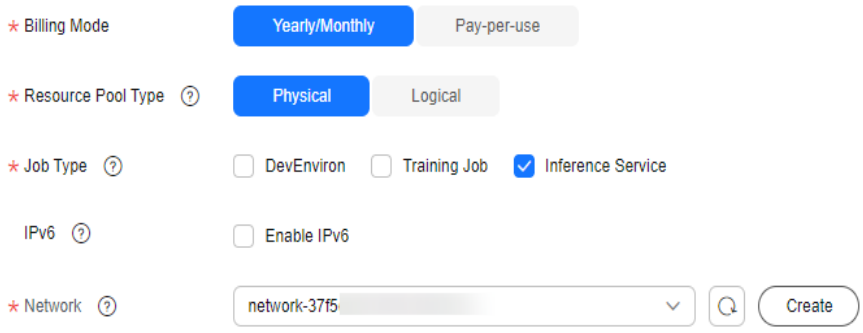


Figura 5-8 Interconexión de las VPC

La interconexión de una VPC permite que el grupo de recursos de ModelArts intercambie datos con su VPC. Antes de conectar la VPC, cree una VPC y una subred. Para obtener más detalles, consulte **Creación de una VPC y una subred**.

Paso 2 Instale y configure un proxy de reenvío para su VPC.

Antes de instalar un proxy de reenvío, compre un ECS con la última imagen de Ubuntu y vincule una EIP al ECS. Luego, inicie sesión en el ECS, instale y configurar un proxy de reenvío de squid.

1. Si Docker no está instalado, ejecute el siguiente comando para instalarlo:
   ```
   curl -sSL https://get.daocloud.io/docker | sh
   ```
2. Tira de la imagen de squid.
   ```
   docker pull ubuntu/squid
   ```
3. Cree un directorio de host y configure **whitelist.conf** y **squid.conf**.

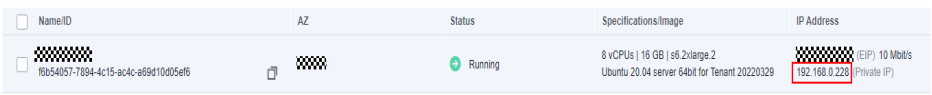
   Create a host directory:
   ```
   mkdir -p /etc/squid/
   ```
   Agregue el archivo de configuración **whitelist.conf**. El contenido son las direcciones a las que se puede acceder. Por ejemplo:
   ```
   .apig.cn-east-3.huaweicloudapis.com
   ```
   Agregue el archivo de configuración **squid.conf**, que incluye lo siguiente:
   ```
   # An ACL named 'whitelist'
   acl whitelist dstdomain '/etc/squid/whitelist.conf'
   # Allow whitelisted URLs through
   http_access allow whitelist
   # Block the rest
   http_access deny all
   # Default port
   http_port 3128
   ```
   Configure los permisos en el directorio del host y en los archivos de configuración:
   ```
   chmod 640 -R /etc/squid
   ```
4. Inicie una instancia de squid.
   ```
   docker run -d --name squid -e TZ=UTC -v /etc/squid:/etc/squid -p 3128:3128
   ubuntu/squid:latest
   ```
5. Si se actualiza **whitelist.conf** o **squid.conf**, vaya al contenedor y actualice el squid.
   ```
   docker exec -it squid bash
   root@{container_id}:/# squid -k reconfigure
   ```

Paso 3 Configure el proxy de DNS y la URL de acceso a Internet en la imagen del algoritmo.

1. Configure el proxy.

   En el código, especifique la dirección IP privada y el puerto del servidor proxy, como se muestra a continuación:
   ```
   proxies = {
     "http": "http://{proxy_server_private_ip}:3128",
     "https": "http://{proxy_server_private_ip}:3128"
   }
   ```
   La siguiente figura muestra cómo obtener la dirección IP privada de un servidor.

   Figura 5-9 Dirección IP privada

2. Configure la URL de acceso a Internet.

   En el código de inferencia, utilice la dirección de URL del servicio para enviar una solicitud de servicio, por ejemplo:
   ```
   https://e8a048ce25136addbbac23ce6132a.apig.cn-east-3.huaweicloudapis.com
   ```

----Fin

5.3 O&M de extremo a extremo para servicios de inferencia

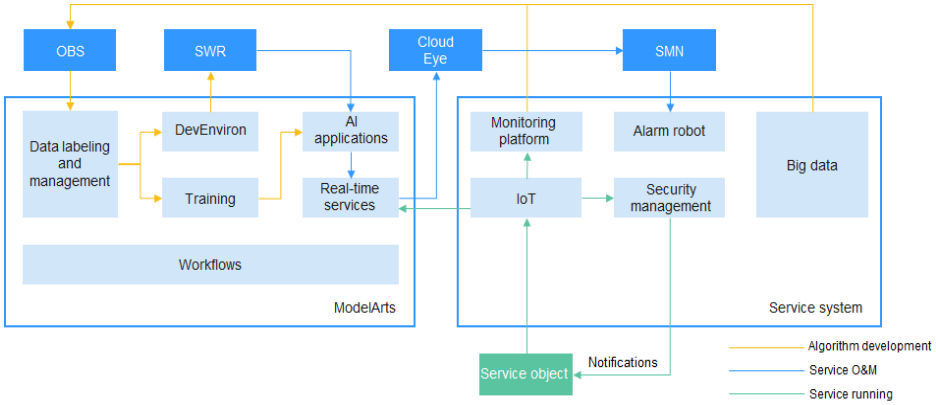
El O&M de extremo a extremo de los servicios de inferencia de ModelArts implica todo el proceso de IA, incluido el desarrollo de algoritmos, el O&M del servicio y la ejecución del servicio.

Descripción general

Proceso de O&M de extremo a extremo

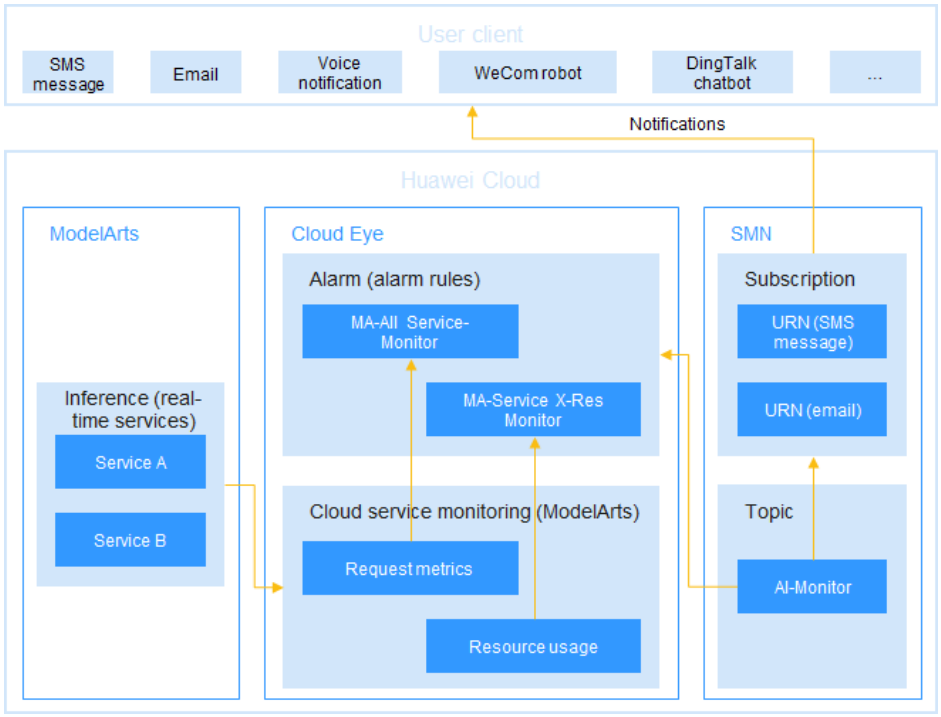
- Durante el desarrollo del algoritmo, almacene los datos del servicio en Object Storage Service (OBS) y luego etiquete y gestione los datos mediante la gestión de datos de ModelArts. Después de entrenar los datos, obtenga un modelo de IA y cree imágenes de aplicaciones de IA con un entorno de desarrollo.
- Durante el O&M de servicio, utilice una imagen para crear una aplicación de IA y despliegar la aplicación de IA como un servicio en tiempo real. Puede obtener los datos de monitoreo del servicio de ModelArts en tiempo real en la consola de gestión de Cloud Eye. Configure las reglas de alarmas para que pueda recibir notificaciones de alarmas en tiempo real.
- Durante la ejecución del servicio, acceda a las solicitudes de servicio en tiempo real en el sistema de servicio y luego configure la lógica y el monitoreo del servicio.

Figura 5-10 Proceso de O&M de extremo a extremo para servicios de inferencia



Durante todo el proceso de O&M, se monitorean las fallas en las solicitudes de servicio y el alto uso de recursos. Cuando se alcance el umbral de uso de recursos, el sistema le enviará una notificación de alarma.

Figura 5-11 Proceso de alarma



Ventajas

El O&M de servicio de extremo a extremo le permite verificar fácilmente el funcionamiento de los servicios tanto en horas punta como en horas valle y detectar el estado de salud de los servicios en tiempo real.

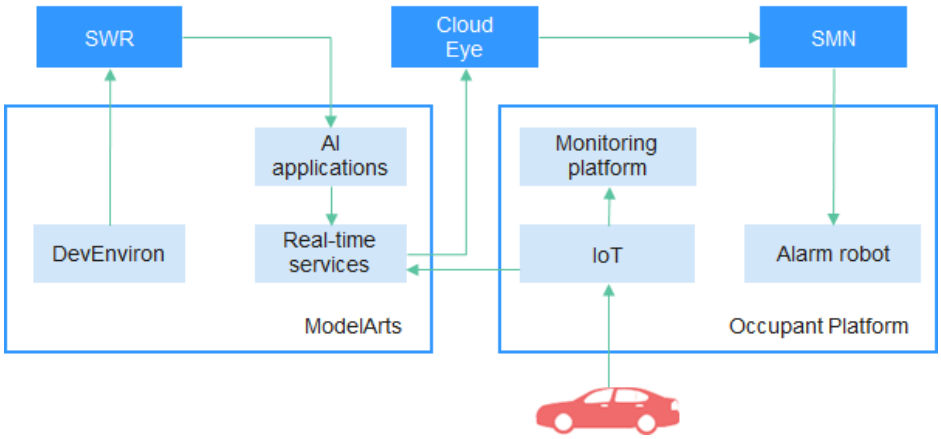
Restricciones

Cloud Eye no supervisa los servicios de inferencia por lotes o de borde.

Procedimiento

Esta sección utiliza un algoritmo de seguridad de los ocupantes en viajes como ejemplo para describir cómo utilizar ModelArts para desplegar y actualizar servicios basados en procesos, así como para O&M y monitoreo automáticos de servicios.

Figura 5-12 Implementación del algoritmo de seguridad de los ocupantes



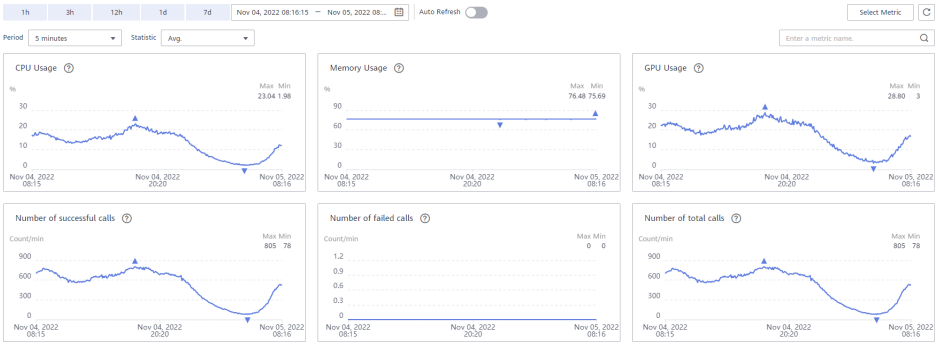
Paso 1 Utilice un modelo desarrollado localmente para crear una imagen personalizada y utilice la imagen para crear una aplicación de AI en ModelArts. Para obtener más detalles, consulte **Creación de una imagen personalizada y su uso para crear una aplicación de IA**.

Paso 2 En la consola de gestión de ModelArts, despliegue la aplicación de IA creada como un servicio en tiempo real.

Paso 3 Inicie sesión en la consola de gestión de Cloud Eye, configure las reglas de alarmas de ModelArts y habilite notificaciones con un tema relacionado. Para obtener más detalles, consulte **Configuración de reglas de alarma**.

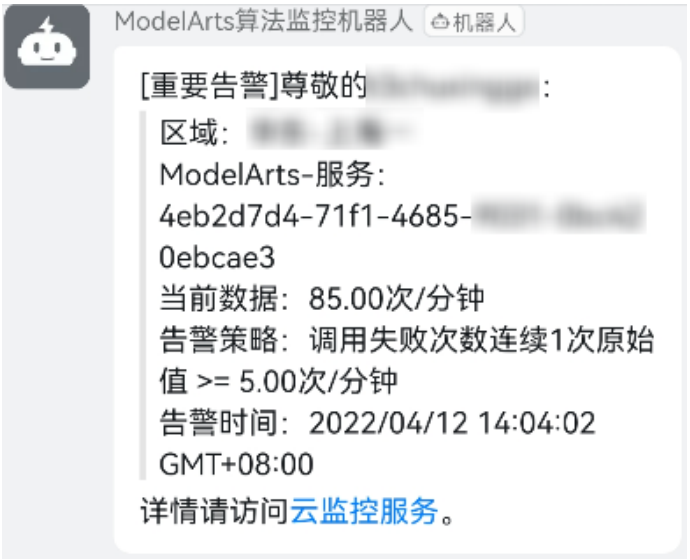
Después de la configuración, seleccione **Cloud Service Monitoring** > **ModelArts** en el panel de navegación de la izquierda para ver las solicitudes y el uso de recursos del servicio en tiempo real.

Figura 5-13 Consulta de las métricas de supervisión del servicio



Cuando se activa una alarma basada en los datos supervisados, el objeto que se ha suscrito al tema de destino recibirá una notificación de mensaje.

Figura 5-14 Alarm-triggered message notification



----Fin

5.4 Creación de una aplicación de IA con un motor personalizado

Cuando utiliza un motor personalizado para crear una aplicación de IA, puede seleccionar la imagen almacenada en SWR como motor y especificar un directorio de archivos en OBS como paquete de modelos. De este modo, puede utilizar imágenes propias para satisfacer sus necesidades específicas.

Antes de desplegar una aplicación de IA como servicio, ModelArts descarga la imagen de SWR en el clúster e inicia la imagen como un contenedor como usuario cuyo UID es 1000 y GID es 100. Luego, ModelArts descarga el archivo de OBS en el directorio **/home/mind/model** del contenedor y ejecuta el comando de arranque preestablecido en la imagen de SWR. El servicio disponible para el puerto 8080 del contenedor se registra automáticamente con APIG. Puede acceder al servicio con el URL APIG.

Especificaciones para utilizar un motor personalizado para crear una aplicación de IA

Para utilizar un motor personalizado a crear una aplicación de IA, asegúrese de que la imagen de SWR, el paquete del modelo de OBS y el tamaño del archivo cumplan con los siguientes requisitos:

- Especificaciones de imagen de SWR
  - Se debe incorporar un usuario común denominado **ma-user** en el grupo **ma-group** en la imagen de SWR. Adicionalmente, el UID y el GID del usuario deben ser 1000 y 100, respectivamente. El siguiente es el comando de dockerfile para el usuario integrado:
    ```
    groupadd -g 100 ma-group && useradd -d /home/ma-user -m -u 1000 -g 100 -s /bin/bash ma-user
    ```
  - Especifique un comando para iniciar la imagen. Especifique **cmd** en el dockerfile. A continuación se muestra un ejemplo:
    ```
    CMD sh /home/mind/run.sh
    ```
    Personalice el archivo de entrada de inicio **run.sh**. Lo siguiente es un ejemplo.
    ```
    #!/bin/bash

    # User-defined script content
    ...

    # run.sh calls app.py to start the server. For details about app.py, see "HTTPS Example".
    python app.py
    ```
  - El servicio debe estar habilitado para HTTPS y está disponible en el puerto 8080. Para obtener más detalles, consulte el **ejemplo de HTTPS**.
  - (Opcional) En el puerto 8080, active la comprobación de estado con la URL**/health**. (La dirección URL de comprobación de estado debe ser **/health**)
- Especificaciones del paquete del modelo de OBS

  El nombre del paquete de modelos debe ser **model**. Para obtener más detalles sobre las especificaciones de paquetes de modelo, consulte **Introducción a especificaciones de paquetes de modelo**.
- Especificaciones de tamaño de archivo

Cuando se utiliza un grupo de recursos públicos, el tamaño total de la imagen de SWR (no la imagen comprimida que se muestra en la página de SWR) y el paquete del modelo de OBS no puede exceder los 30 GB.

Ejemplo de HTTPS

Utilice Flask para iniciar HTTPS. A continuación se muestra un ejemplo del código del servidor web:

```
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))

@app.route('/health', methods=['GET'])
def healthy():
    return "{\"status\": \"OK\"}"

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

Depuración en un equipo local

Realice las siguientes operaciones en una computadora local con Docker instalado para verificar si un motor personalizado cumple con las especificaciones:

1. Descargue la imagen personalizada, por ejemplo, **custom_engine:v1** en la computadora local.
2. Copie la carpeta del paquete de modelo **model** en el host local.
3. Ejecute el siguiente comando en el mismo directorio que la carpeta del paquete modelo para iniciar el servicio:
   ```
   docker run --user 1000:100 -p 8080:8080 -v model:/home/mind/model
   custom_engine:v1
   ```

   NOTA

   Este comando solo se utiliza para simulaciones porque el directorio montado en **-v** tiene asignado el permiso de root. En el entorno de nube, después de descargar el archivo de modelo desde OBS a **/home/mind/model**, el propietario del archivo se cambia a **ma-user**.
4. Inicie otro terminal en el equipo local y ejecute el siguiente comando para obtener el resultado de inferencia esperado:


```
curl https://127.0.0.1:8080/${Request path to the inference service}
```

Ejemplo de despliegue

En la siguiente sección se describe cómo utilizar un motor personalizado para crear una aplicación de IA.

1. Cree una aplicación de IA y vea sus detalles.

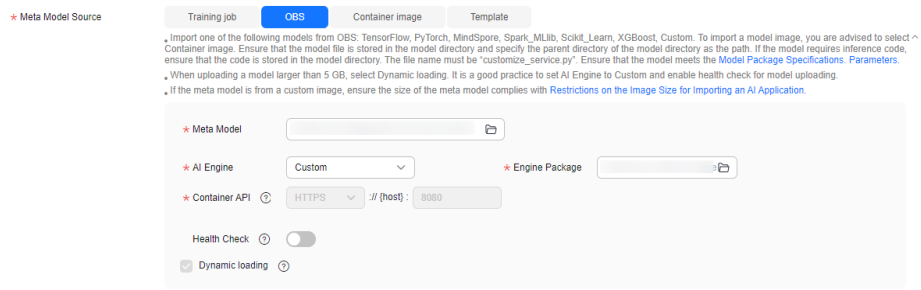
   Inicie sesión en la consola de ModelArts, seleccione **AI Application Management** > **AI Applications** y haga clic en **Create**. En la página que aparece, configure los siguientes parámetros:

   – **Meta Model Source**: **OBS**
   – **Meta Model**: un paquete de modelo seleccionado de OBS
   – **AI Engine**: **Custom**
   – **Engine Package**: una imagen de SWR

     Mantenga la configuración predeterminada para otros parámetros.

   Haga clic en **Create Now**. En la lista de aplicaciones de IA que aparece en pantalla, compruebe el estado de la aplicación de IA. Cuando su estado cambia a **Normal**, se crea la aplicación de IA.

   Figura 5-15 Creación de una aplicación de IA

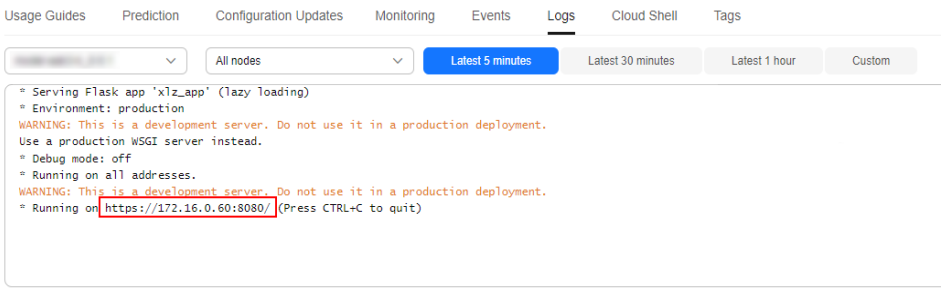
   Haga clic en el nombre de la aplicación de IA. En la página que aparece, vea detalles sobre la aplicación de IA.

2. Despliegue la aplicación de IA como servicio y consulte los detalles del servicio.

   En la página de detalles de la aplicación de IA, seleccione **Deploy** > **Real-Time Services** en la esquina superior derecha. En la página **Deploy**, seleccione una especificación de nodo de cómputo adecuada (por ejemplo, **CPU: 2 vCPUs 8 GB**), mantenga la configuración por defecto de otros parámetros y haga clic en **Next**. Cuando el estado del servicio cambia a **Running**, se despliegue ese servicio.

   Haga clic en el nombre del servicio. En la página que aparece, vea los detalles del servicio. Haga clic en la pestaña **Logs** para ver los logs de servicio.

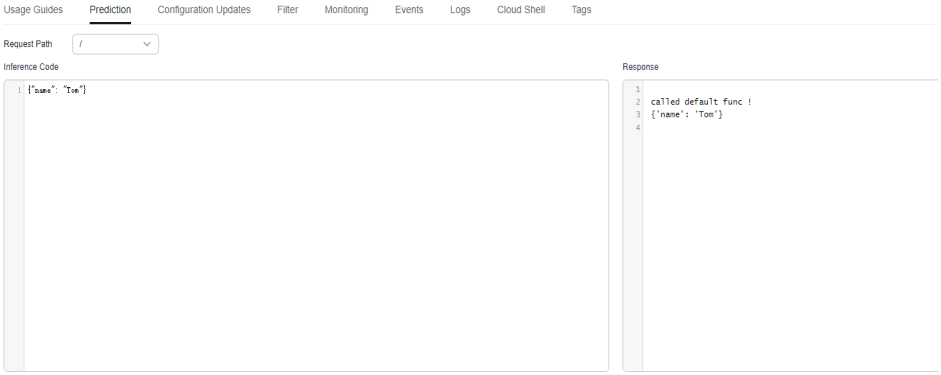
Figura 5-16 Logs



3. Utilice el servicio para predicciones.

   En la página de detalles del servicio, haga clic en la ficha **Prediction** para utilizar el servicio de predicción.

Figura 5-17 Predicción



5.5 Creación de una aplicación de IA con un modelo grande y despliegue de un servicio en tiempo real

Contexto

Actualmente, un gran modelo puede tener cientos de miles de millones o incluso billones de parámetros, y su tamaño es cada vez mayor. Un modelo grande con cientos de miles de millones de parámetros supera los 200 GB, y plantea nuevos requisitos para la gestión de versiones y el despliegue en producción de la plataforma. Por ejemplo, las aplicaciones de IA requieren un ajuste dinámico de la cuota de almacenamiento del tenant. La lentitud de carga y arranque de los modelos requiere una configuración flexible del tiempo de espera en el despliegue. Es necesario acortar el tiempo de recuperación del servicio en caso de que sea necesario volver a cargar el modelo tras un reinicio provocado por una excepción de carga.

Para satisfacer los requisitos anteriores, la plataforma de inferencia de ModelArts ofrece una solución para la gestión de aplicaciones de IA y el despliegue de servicios en los escenarios de aplicaciones de modelos grandes.

Restricciones

- Debe solicitar la cuota de tamaño de una aplicación de IA y agregar la lista blanca almacenada en caché utilizando el almacenamiento local del nodo.

- Debe utilizar el motor personalizado **Custom** para configurar la carga dinámica.
- Se requiere un grupo de recursos dedicado para desplegar el servicio.
- El espacio en disco del grupo de recursos dedicado debe ser superior a 1 TB.

Procedimiento

1. **Aplicación para aumentar la cuota de tamaño de una aplicación de IA y utilizar el almacenamiento local del nodo para almacenar en caché la lista blanca**
2. **Carga de datos del modelo y verificación de la coherencia de los objetos cargados**
3. **Cree un grupo de recursos dedicado**
4. **Creación de una aplicación de IA**
5. **Despliegue de un servicio en tiempo real**

Aplicación para aumentar la cuota de tamaño de una aplicación de IA y utilizar el almacenamiento local del nodo para almacenar en caché la lista blanca

Durante el despliegue del servicio, el paquete del modelo cargado dinámicamente se almacena en el espacio temporal del disco de forma predeterminada. Cuando se detiene el servicio, los archivos cargados se eliminan y deben volver a cargarse cuando se reinicia el servicio. Para evitar la carga repetida, la plataforma permite cargar el paquete modelo desde el espacio de almacenamiento local del nodo en el grupo de recursos y mantiene válidos los archivos cargados incluso cuando el servicio se detiene o se reinicia (utilizando el valor hash para garantizar la coherencia de los datos).

Para utilizar un modelo de gran tamaño, es necesario utilizar un motor personalizado y activar la carga dinámica al importar el modelo. En este sentido, debe realizar las siguientes operaciones:

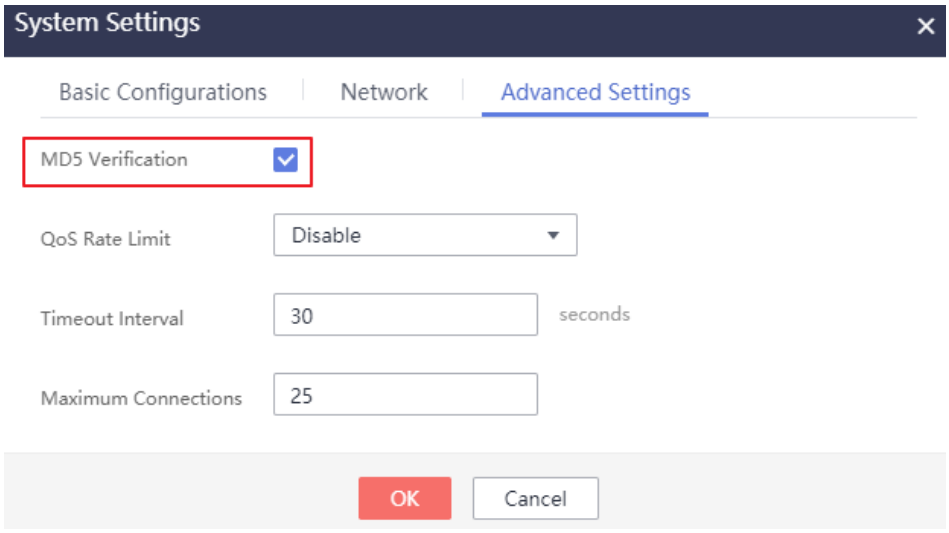
- Si el tamaño del modelo supera la cuota predeterminada, envíe un ticket de servicio para aumentar la cuota de tamaño de una sola aplicación de IA. La cuota de tamaño predeterminada de una aplicación de IA es de 20 GB.
- Envíe un ticket de servicio para agregar la lista blanca almacenada en caché utilizando el almacenamiento local del nodo.

Carga de datos del modelo y verificación de la coherencia de los objetos cargados

Para garantizar la integridad de los datos durante la carga dinámica, es necesario verificar la coherencia de los objetos cargados al cargar los datos del modelo en OBS. Los SDK de obsutil, OBS Browser+ y OBS soportan la verificación de la consistencia de los datos durante la carga. Puede elegir el método que mejor se adapte a sus necesidades. Para obtener más detalles, consulte **Verificación de la consistencia de los datos durante la carga**.

Por ejemplo, si carga datos con OBS Browser+, active la verificación de MD5, como se muestra en **Figura 5-18**. Cuando la carga dinámica está activada y se utiliza el almacenamiento persistente local del nodo, OBS Browser+ verifica la consistencia de los datos durante la carga de datos.

Figura 5-18 Configuración de la verificación de MD5 para OBS Browser+



Creación de un grupo de recursos dedicado

Para utilizar el almacenamiento local persistente, debe crear un grupo de recursos dedicado cuyo espacio en disco sea superior a 1 TB. Puede ver la información del disco en la ficha **Specifications** de la página **Basic Information** del grupo de recursos dedicado. Si un servicio no puede desplegarse y el sistema muestra un mensaje indicando que el espacio en disco es insuficiente, consulte **¿Qué debo hacer si los recursos son insuficientes cuando se despliega, inicia, actualiza o modifica un servicio en tiempo real?**.

Figura 5-19 Consulta de la información de disco del grupo de recursos dedicado

Creación de una aplicación de IA

Si utiliza un modelo grande para crear una aplicación de IA e importar el modelo desde OBS, complete las siguientes configuraciones:

1. Utilice un motor personalizado y active la carga dinámica.

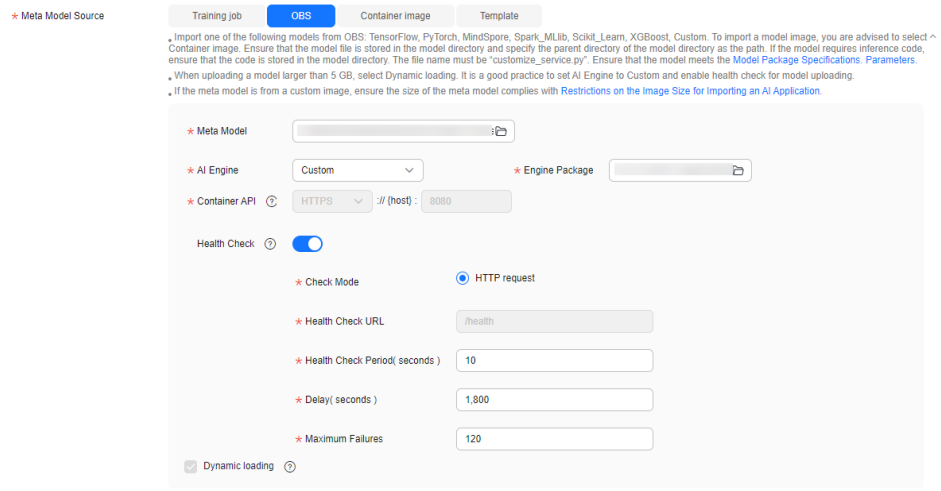
   Para utilizar un modelo de gran tamaño, es necesario utilizar un motor personalizado y activar la carga dinámica al importar el modelo. Puede crear un motor personalizado para satisfacer requisitos especiales de paquetes de dependencias de imágenes y marcos de inferencia en escenarios de modelos grandes. Consulte **Creación de una aplicación de IA con un motor personalizado** para obtener detalles sobre cómo crear un motor personalizado.

   Cuando se utiliza un motor personalizado, la carga dinámica está activada de forma predeterminada. El paquete de modelo se separa de la imagen y el modelo se carga dinámicamente en la carga de servicio durante el despliegue de servicio.

2. Configure la comprobación de estado.

   La comprobación de la salud es obligatoria para las aplicaciones de IA importadas utilizando un gran modelo para identificar los servicios no disponibles que se muestran como iniciados.

Figura 5-20 Uso de un motor personalizado, activación de la carga dinámica y configuración de la comprobación de estado



Despliegue de un servicio en tiempo real

Al desplegar el servicio, complete las siguientes configuraciones:

1. Personalice el intervalo de tiempo de espera de despliegue.

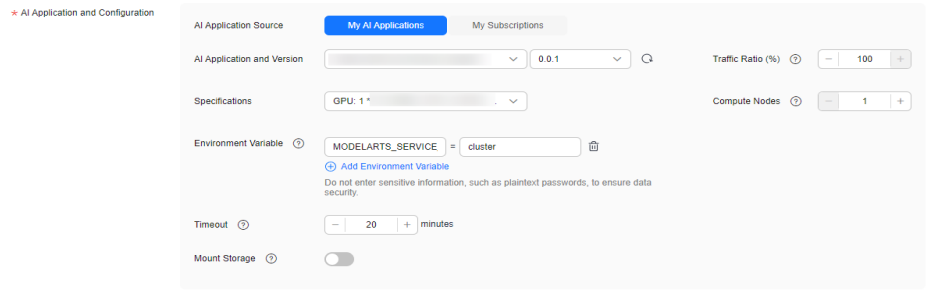
   Por lo general, el tiempo necesario para cargar e iniciar un modelo grande es largo al que el de un modelo común. Establezca **Timeout** en un valor adecuado. De lo contrario, el tiempo de espera puede transcurrir antes de que finalice la puesta en marcha del modelo, y el despliegue puede fallar.

2. Agregue una variable de entorno.

   Durante el despliegue del servicio, agregue la siguiente variable de entorno para establecer la política de equilibrio de carga del tráfico de servicio en afinidad de clúster, evitando que las instancias de servicio no preparadas afecten a la tasa de éxito de la predicción:

   ```
   MODELARTS_SERVICE_TRAFFIC_POLICY: cluster
   ```

Figura 5-21 Personalización del intervalo de despliegue y adición de una variable de entorno



Se recomienda desplegar varias instancias para mejorar la confiabilidad del servicio.

5.6 Migración de un marco de inferencia de terceros a un motor de inferencia personalizado

Contexto

ModelArts permite el despliegue de marcos de inferencia de terceros. Esta sección describe cómo migrar TF Serving y Triton a un motor de inferencia personalizado.

- TensorFlow Serving (TF Serving) es un sistema de despliegue de modelo flexible y de alto rendimiento para el aprendizaje automático. Proporciona funciones de gestión de versiones de modelos y de reversión de servicios. Al configurar parámetros como la ruta del modelo, el puerto del modelo y el nombre del modelo, las imágenes nativas de TF Serving pueden comenzar rápidamente a proporcionar servicios a los que se puede acceder con las API gRPC y HTTP RESTful.
- Triton es un marco de servicios de inferencia de alto rendimiento desarrollado por NVIDIA. Soporta múltiples protocolos de servicio, incluidos HTTP y gRPC. Además, Triton es compatible con varios motores de inferencia como TensorFlow, TensorRT, PyTorch y ONNX Runtime. En concreto, permite la concurrencia multimodelo y el procesamiento dinámico por lotes, lo que optimiza la utilización de la GPU y mejora el rendimiento del servicio de inferencia.

La migración de un marco de terceros a un marco de inferencia de ModelArts requiere la reconstrucción de la imagen nativa del marco de terceros. Después de eso, se puede utilizar la gestión de versiones del modelo de ModelArts y la carga dinámica del modelo. En esta sección se muestra cómo completar una reconstrucción de este tipo. Después de crear una imagen del motor personalizado, puede usarla para crear una versión de aplicación de IA y desplegar y gestionar servicios con la aplicación de IA.

La siguiente figura muestra los elementos de reconstrucción.

Figura 5-22 Elementos de reconstrucción

El proceso de reconstrucción puede diferir para las imágenes de diversos marcos. Para obtener más detalles, consulte el procedimiento de migración específico del marco de destino.

- **Migración de TF Serving**
- **Migración de Triton**

Migración de TF Serving

Paso 1 Agregue el usuario **ma-user**.

La imagen se crea según la imagen nativa **tensorflow/serving:2.8.0**. El grupo de usuarios **100** existe en la imagen de forma predeterminada. Ejecute el siguiente comando en Dockerfile para agregar usuario **ma-user**:


```
RUN useradd -d /home/ma-user -m -u 1000 -g 100 -s /bin/bash ma-user
```

Paso 2 Configure un proxy de Nginx para soportar HTTPS.

Después de convertir el protocolo a HTTPS, el puerto expuesto cambia de 8501 de TF Serving a 8080.

1. Ejecute los siguientes comandos en Dockerfile para instalar y configurar Nginx:

```
RUN apt-get update && apt-get -y --no-install-recommends install nginx && apt-get clean
RUN mkdir /home/mind && \
    mkdir -p /etc/nginx/keys && \
    mkfifo /etc/nginx/keys/fifo && \
    chown -R ma-user:100 /home/mind && \
    rm -rf /etc/nginx/conf.d/default.conf && \
    chown -R ma-user:100 /etc/nginx/ && \
    chown -R ma-user:100 /var/log/nginx && \
    chown -R ma-user:100 /var/lib/nginx && \
    sed -i "s#/var/run/nginx.pid#/home/ma-user/nginx.pid#g" /etc/init.d/nginx
ADD nginx /etc/nginx
ADD run.sh /home/mind/
ENTRYPOINT []
CMD /bin/bash /home/mind/run.sh
```

2. Cree el directorio de Nginx.

```
nginx
├──nginx.conf
└──conf.d
       ├── modelarts-model-server.conf
```

3. Escriba el archivo nginx.conf.

```
user ma-user 100;
worker_processes 2;
pid /home/ma-user/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;
events {
    worker_connections 768;
}
http {
    ##
    # Basic Settings
    ##
    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    types_hash_max_size 2048;
    fastcgi_hide_header X-Powered-By;
    port_in_redirect off;
    server_tokens off;
    client_body_timeout 65s;
    client_header_timeout 65s;
    keepalive_timeout 65s;
    send_timeout 65s;
    # server_names_hash_bucket_size 64;
    # server_name_in_redirect off;
    include /etc/nginx/mime.types;
    default_type application/octet-stream;
    ##
    # SSL Settings
    ##
    ssl_protocols TLSv1.2;
    ssl_prefer_server_ciphers on;
    ssl_ciphers ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256;
    ##
    # Logging Settings
    ##
    access_log /var/log/nginx/access.log;
    error_log /var/log/nginx/error.log;
    ##
    # Gzip Settings
```



```
    ##
    gzip on;
    ##
    # Virtual Host Configs
    ##
    include /etc/nginx/conf.d/modelarts-model-server.conf;
}
```

4. Escriba el archivo de configuración modelarts-model-server.conf.

```
server {
    client_max_body_size 15M;
    large_client_header_buffers 4 64k;
    client_header_buffer_size 1k;
    client_body_buffer_size 16k;
    ssl_certificate /etc/nginx/ssl/server/server.crt;
    ssl_password_file /etc/nginx/keys/fifo;
    ssl_certificate_key /etc/nginx/ssl/server/server.key;
    # setting for mutual ssl with client
    ##
    # header Settings
    ##
    add_header X-XSS-Protection "1; mode=block";
    add_header X-Frame-Options SAMEORIGIN;
    add_header X-Content-Type-Options nosniff;
    add_header Strict-Transport-Security "max-age=31536000;
includeSubdomains;";
    add_header Content-Security-Policy "default-src 'self'";
    add_header Cache-Control "max-age=0, no-cache, no-store, must-revalidate";
    add_header Pragma "no-cache";
    add_header Expires "-1";
    server_tokens off;
    port_in_redirect off;
    fastcgi_hide_header X-Powered-By;
    ssl_session_timeout 2m;
    ##
    # SSL Settings
    ##
    ssl_protocols TLSv1.2;
    ssl_prefer_server_ciphers on;
    ssl_ciphers ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256;
    listen    0.0.0.0:8080 ssl;
    error_page 502 503 /503.html;
    location /503.html {
        return 503 '{"error_code": "ModelArts.4503","error_msg": "Failed to
connect to backend service, please confirm your service is connectable. "}';
    }
    location / {
#       limit_req zone=mylimit;
#       limit_req_status 429;
        proxy_pass http://127.0.0.1:8501;
    }
}
```

5. Cree un script de inicio.

   NOTA

   Antes de ejecutar el script de inicio de TF Serving, debe crear un certificado de SSL.

   El código de ejemplo del script de inicio **run.sh** es el siguiente:

```
#!/bin/bash
mkdir -p /etc/nginx/ssl/server && cd /etc/nginx/ssl/server
cipherText=$(openssl rand -base64 32)
openssl genrsa -aes256 -passout pass:"${cipherText}" -out server.key 2048
openssl rsa -in server.key -passin pass:"${cipherText}" -pubout -out
rsa_public.key
openssl req -new -key server.key -passin pass:"${cipherText}" -out server.csr
-subj "/C=CN/ST=GD/L=SZ/O=Huawei/OU=ops/CN=*.huawei.com"
openssl genrsa -out ca.key 2048
openssl req -new -x509 -days 3650 -key ca.key -out ca-crt.pem -subj "/C=CN/
```



```
    ST=GD/L=SZ/O=Huawei/OU=dev/CN=ca"
    openssl x509 -req -days 3650 -in server.csr -CA ca-crt.pem -CAkey ca.key -
    CAcreateserial -out server.crt
    service nginx start &
    echo ${cipherText} > /etc/nginx/keys/fifo
    unset cipherText
    sh /usr/bin/tf_serving_entrypoint.sh
```

Paso 3 Modifique la ruta del modelo por defecto para admitir la carga dinámica del modelo de ModelArts.

Ejecute los siguientes comandos en Dockerfile para cambiar la ruta predeterminada del modelo:

```
ENV MODEL_BASE_PATH /home/mind
ENV MODEL_NAME model
```

----Fin

Ejemplo de Dockerfile:

```
FROM tensorflow/serving:2.8.0
RUN useradd -d /home/ma-user -m -u 1000 -g 100 -s /bin/bash ma-user
RUN apt-get update && apt-get -y --no-install-recommends install nginx && apt-get
clean
RUN mkdir /home/mind && \
    mkdir -p /etc/nginx/keys && \
    mkfifo /etc/nginx/keys/fifo && \
    chown -R ma-user:100 /home/mind && \
    rm -rf /etc/nginx/conf.d/default.conf && \
    chown -R ma-user:100 /etc/nginx/ && \
    chown -R ma-user:100 /var/log/nginx && \
    chown -R ma-user:100 /var/lib/nginx && \
    sed -i "s#/var/run/nginx.pid#/home/ma-user/nginx.pid#g" /etc/init.d/nginx
ADD nginx /etc/nginx
ADD run.sh /home/mind/
ENV MODEL_BASE_PATH /home/mind
ENV MODEL_NAME model
ENTRYPOINT []
CMD /bin/bash /home/mind/run.sh
```

Migración de Triton

Esta sección utiliza la imagen **nvcr.io/nvidia/tritonserver:23.03-py3** proporcionada por NVIDIA para la adaptación y el modelo de base de código abierto LLaMA 7B para la inferencia.

Paso 1 Agregue el usuario **ma-user**.

El usuario **triton-server**, cuyo ID es 1000, existe en la imagen de Triton de forma predeterminada. Cambie el ID de usuario **triton-server** y agregue el usuario **ma-user** ejecutando este comando en Dockerfile.

```
RUN usermod -u 1001 triton-server && useradd -d /home/ma-user -m -u 1000 -g 100 -
s /bin/bash ma-user
```

Paso 2 Configure un proxy de Nginx para soportar HTTPS.

1. Ejecute los siguientes comandos en Dockerfile para instalar y configurar Nginx:

```
RUN apt-get update && apt-get -y --no-install-recommends install nginx && apt-
get clean && \
    mkdir /home/mind && \
    mkdir -p /etc/nginx/keys && \
    mkfifo /etc/nginx/keys/fifo && \
    chown -R ma-user:100 /home/mind && \
    rm -rf /etc/nginx/conf.d/default.conf && \
```



```
    chown -R ma-user:100 /etc/nginx/ && \
    chown -R ma-user:100 /var/log/nginx && \
    chown -R ma-user:100 /var/lib/nginx && \
    sed -i "s#/var/run/nginx.pid#/home/ma-user/nginx.pid#g" /etc/init.d/nginx
```

2. Cree el directorio de Nginx de la siguiente manera:

```
nginx
├──nginx.conf
└──conf.d
       ├── modelarts-model-server.conf
```

3. Escriba el archivo nginx.conf.

```
user ma-user 100;
worker_processes 2;
pid /home/ma-user/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;
events {
    worker_connections 768;
}
http {
    ##
    # Basic Settings
    ##
    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    types_hash_max_size 2048;
    fastcgi_hide_header X-Powered-By;
    port_in_redirect off;
    server_tokens off;
    client_body_timeout 65s;
    client_header_timeout 65s;
    keepalive_timeout 65s;
    send_timeout 65s;
    # server_names_hash_bucket_size 64;
    # server_name_in_redirect off;
    include /etc/nginx/mime.types;
    default_type application/octet-stream;
    ##
    # SSL Settings
    ##
    ssl_protocols TLSv1.2;
    ssl_prefer_server_ciphers on;
    ssl_ciphers ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256;
    ##
    # Logging Settings
    ##
    access_log /var/log/nginx/access.log;
    error_log /var/log/nginx/error.log;
    ##
    # Gzip Settings
    ##
    gzip on;
    ##
    # Virtual Host Configs
    ##
    include /etc/nginx/conf.d/modelarts-model-server.conf;
}
```

4. Escriba el archivo de configuración modelarts-model-server.conf.

```
server {
    client_max_body_size 15M;
    large_client_header_buffers 4 64k;
    client_header_buffer_size 1k;
    client_body_buffer_size 16k;
    ssl_certificate /etc/nginx/ssl/server/server.crt;
    ssl_password_file /etc/nginx/keys/fifo;
    ssl_certificate_key /etc/nginx/ssl/server/server.key;
    # setting for mutual ssl with client
    ##
    # header Settings
```



```
    ##
    add_header X-XSS-Protection "1; mode=block";
    add_header X-Frame-Options SAMEORIGIN;
    add_header X-Content-Type-Options nosniff;
    add_header Strict-Transport-Security "max-age=31536000;
includeSubdomains;";
    add_header Content-Security-Policy "default-src 'self'";
    add_header Cache-Control "max-age=0, no-cache, no-store, must-revalidate";
    add_header Pragma "no-cache";
    add_header Expires "-1";
    server_tokens off;
    port_in_redirect off;
    fastcgi_hide_header X-Powered-By;
    ssl_session_timeout 2m;
    ##
    # SSL Settings
    ##
    ssl_protocols TLSv1.2;
    ssl_prefer_server_ciphers on;
    ssl_ciphers ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256;
    listen    0.0.0.0:8080 ssl;
    error_page 502 503 /503.html;
    location /503.html {
        return 503 '{"error_code": "ModelArts.4503","error_msg": "Failed to
connect to backend service, please confirm your service is connectable. "}';
    }
    location / {
#       limit_req zone=mylimit;
#       limit_req_status 429;
        proxy_pass http://127.0.0.1:8000;
    }
}
```

5. Cree un script de inicio **run.sh**.

   NOTA

   Antes de ejecutar el script de inicio de Triton, debe crear un certificado de SSL.

```
#!/bin/bash
mkdir -p /etc/nginx/ssl/server && cd /etc/nginx/ssl/server
cipherText=$(openssl rand -base64 32)
openssl genrsa -aes256 -passout pass:"${cipherText}" -out server.key 2048
openssl rsa -in server.key -passin pass:"${cipherText}" -pubout -out
rsa_public.key
openssl req -new -key server.key -passin pass:"${cipherText}" -out server.csr
-subj "/C=CN/ST=GD/L=SZ/O=Huawei/OU=ops/CN=*.huawei.com"
openssl genrsa -out ca.key 2048
openssl req -new -x509 -days 3650 -key ca.key -out ca-crt.pem -subj "/C=CN/
ST=GD/L=SZ/O=Huawei/OU=dev/CN=ca"
openssl x509 -req -days 3650 -in server.csr -CA ca-crt.pem -CAkey ca.key -
CAcreateserial -out server.crt
service nginx start &
echo ${cipherText} > /etc/nginx/keys/fifo
unset cipherText

bash /home/mind/model/triton_serving.sh
```

Paso 3 Prepare **tensorrtllm_backend**.

1. Obtener el código de fuente de **tensorrtllm_backend**; instalar las dependencias (TensorRT, CMake y PyTorch); compilar e instalar.

```
# get tensortllm_backend source code
WORKDIR /opt/tritonserver
RUN apt-get install -y --no-install-recommends rapidjson-dev python-is-
python3 git-lfs && \
    git config --global http.sslVerify false && \
    git config --global http.postBuffer 1048576000 && \
    git clone -b v0.5.0 https://github.com/triton-inference-server/
tensorrtllm_backend.git --depth 1 && \
```



```
    cd tensorrtllm_backend && git lfs install && \
    git config submodule.tensorrt_llm.url https://github.com/NVIDIA/TensorRT-
LLM.git && \
    git submodule update --init --recursive --depth 1 && \
    pip3 install -r requirements.txt

# build tensorrtllm_backend
WORKDIR /opt/tritonserver/tensorrtllm_backend/tensorrt_llm
RUN sed -i "s/wget/wget --no-check-certificate/g" docker/common/
install_tensorrt.sh && \
    bash docker/common/install_tensorrt.sh && \
    export  LD_LIBRARY_PATH=/usr/local/tensorrt/lib:${LD_LIBRARY_PATH} && \
    sed -i "s/wget/wget --no-check-certificate/g" docker/common/
install_cmake.sh && \
    bash docker/common/install_cmake.sh && \
    export PATH=/usr/local/cmake/bin:$PATH && \
    bash docker/common/install_pytorch.sh pypi && \
    python3 ./scripts/build_wheel.py --trt_root /usr/local/tensorrt && \
    pip install ./build/tensorrt_llm-0.5.0-py3-none-any.whl && \
    rm -f ./build/tensorrt_llm-0.5.0-py3-none-any.whl && \
    cd ../inflight_batcher_llm && bash scripts/build.sh && \
    mkdir /opt/tritonserver/backends/tensorrtllm && \
    cp ./build/libtriton_tensorrtllm.so /opt/tritonserver/backends/
tensorrtllm/ && \
    chown -R ma-user:100 /opt/tritonserver
```

2. Cree el script de inicio **triton_serving.sh** del Triton serving. A continuación se muestra un ejemplo para el modelo de LLaMA:

```
MODEL_NAME=llama_7b
MODEL_DIR=/home/mind/model/${MODEL_NAME}
OUTPUT_DIR=/tmp/llama/7B/trt_engines/fp16/1-gpu/
MAX_BATCH_SIZE=1
export LD_LIBRARY_PATH=/usr/local/tensorrt/lib:${LD_LIBRARY_PATH}

# build tensorrt_llm engine
cd /opt/tritonserver/tensorrtllm_backend/tensorrt_llm/examples/llama
python build.py --model_dir ${MODEL_DIR} \
                --dtype float16 \
                --remove_input_padding \
                --use_gpt_attention_plugin float16 \
                --enable_context_fmha \
                --use_weight_only \
                --use_gemm_plugin float16 \
                --output_dir ${OUTPUT_DIR} \
                --paged_kv_cache \
                --max_batch_size ${MAX_BATCH_SIZE}

# set config parameters
cd /opt/tritonserver/tensorrtllm_backend
mkdir triton_model_repo
cp all_models/inflight_batcher_llm/* triton_model_repo/ -r

python3 tools/fill_template.py -i triton_model_repo/preprocessing/
config.pbtxt tokenizer_dir:$
{MODEL_DIR},tokenizer_type:llama,triton_max_batch_size:$
{MAX_BATCH_SIZE},preprocessing_instance_count:1
python3 tools/fill_template.py -i triton_model_repo/postprocessing/
config.pbtxt tokenizer_dir:$
{MODEL_DIR},tokenizer_type:llama,triton_max_batch_size:$
{MAX_BATCH_SIZE},postprocessing_instance_count:1
python3 tools/fill_template.py -i triton_model_repo/ensemble/config.pbtxt
triton_max_batch_size:${MAX_BATCH_SIZE}
python3 tools/fill_template.py -i triton_model_repo/tensorrt_llm/config.pbtxt
triton_max_batch_size:$
{MAX_BATCH_SIZE},decoupled_mode:False,max_beam_width:1,engine_dir:$
{OUTPUT_DIR},max_tokens_in_paged_kv_cache:2560,max_attention_window_size:2560,
kv_cache_free_gpu_mem_fraction:0.5,exclude_input_in_output:True,enable_kv_cach
e_reuse:False,batching_strategy:V1,max_queue_delay_microseconds:600

# launch tritonserver
```



```
python3 scripts/launch_triton_server.py --world_size 1 --
model_repo=triton_model_repo/
while true; do sleep 10000; done
```

Descripción de algunos parámetros:

- **MODEL_NAME**: nombre de la carpeta de OBS donde se almacena el archivo de ponderación del modelo en formato de Hugging Face.
- **OUTPUT_DIR**: ruta al archivo de modelo convertido por TensorRT-LLM en el contenedor.

El Dockerfile completo es el siguiente:

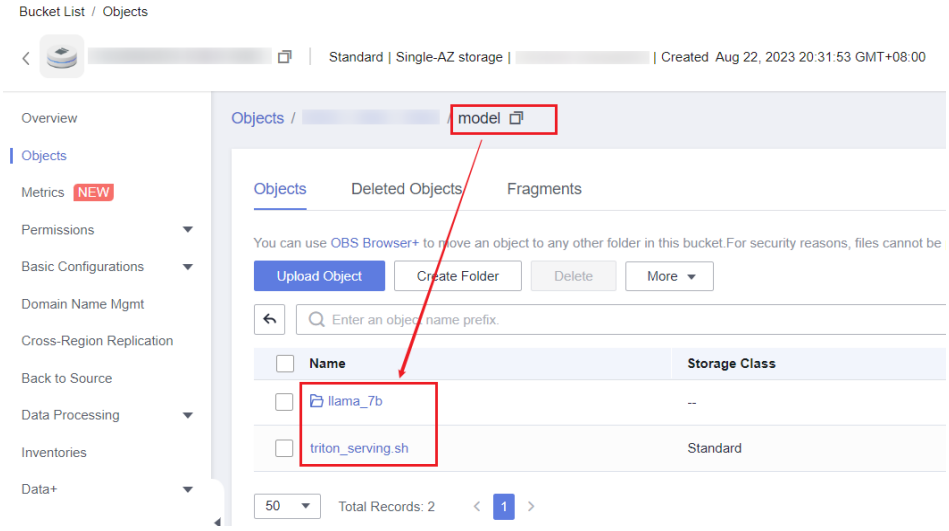
```
FROM nvcr.io/nvidia/tritonserver:23.03-py3

# add ma-user and install nginx
RUN usermod -u 1001 triton-server && useradd -d /home/ma-user -m -u 1000 -g
100 -s /bin/bash ma-user && \
    apt-get update && apt-get -y --no-install-recommends install nginx && apt-
get clean && \
    mkdir /home/mind && \
    mkdir -p /etc/nginx/keys && \
    mkfifo /etc/nginx/keys/fifo && \
    chown -R ma-user:100 /home/mind && \
    rm -rf /etc/nginx/conf.d/default.conf && \
    chown -R ma-user:100 /etc/nginx/ && \
    chown -R ma-user:100 /var/log/nginx && \
    chown -R ma-user:100 /var/lib/nginx && \
    sed -i "s#/var/run/nginx.pid#/home/ma-user/nginx.pid#g" /etc/init.d/nginx

# get tensortllm_backend source code
WORKDIR /opt/tritonserver
RUN apt-get install -y --no-install-recommends rapidjson-dev python-is-
python3 git-lfs && \
    git config --global http.sslVerify false && \
    git config --global http.postBuffer 1048576000 && \
    git clone -b v0.5.0 https://github.com/triton-inference-server/
tensorrtllm_backend.git --depth 1 && \
    cd tensorrtllm_backend && git lfs install && \
    git config submodule.tensorrt_llm.url https://github.com/NVIDIA/TensorRT-
LLM.git && \
    git submodule update --init --recursive --depth 1 && \
    pip3 install -r requirements.txt

# build tensorrtllm_backend
WORKDIR /opt/tritonserver/tensorrtllm_backend/tensorrt_llm
RUN sed -i "s/wget/wget --no-check-certificate/g" docker/common/
install_tensorrt.sh && \
    bash docker/common/install_tensorrt.sh && \
    export  LD_LIBRARY_PATH=/usr/local/tensorrt/lib:${LD_LIBRARY_PATH} && \
    sed -i "s/wget/wget --no-check-certificate/g" docker/common/
install_cmake.sh && \
    bash docker/common/install_cmake.sh && \
    export PATH=/usr/local/cmake/bin:$PATH && \
    bash docker/common/install_pytorch.sh pypi && \
    python3 ./scripts/build_wheel.py --trt_root /usr/local/tensorrt && \
    pip install ./build/tensorrt_llm-0.5.0-py3-none-any.whl && \
    rm -f ./build/tensorrt_llm-0.5.0-py3-none-any.whl && \
    cd ../inflight_batcher_llm && bash scripts/build.sh && \
    mkdir /opt/tritonserver/backends/tensorrtllm && \
    cp ./build/libtriton_tensorrtllm.so /opt/tritonserver/backends/
tensorrtllm/ && \
    chown -R ma-user:100 /opt/tritonserver

ADD nginx /etc/nginx
ADD run.sh /home/mind/
CMD /bin/bash /home/mind/run.sh
```

Una vez creada la imagen, regístrela con Huawei Cloud SWR para desplegar servicios de inferencia en ModelArts.

Paso 4 Utilice la imagen adaptada para desplegar de un servicio de inferencia en tiempo real en ModelArts.

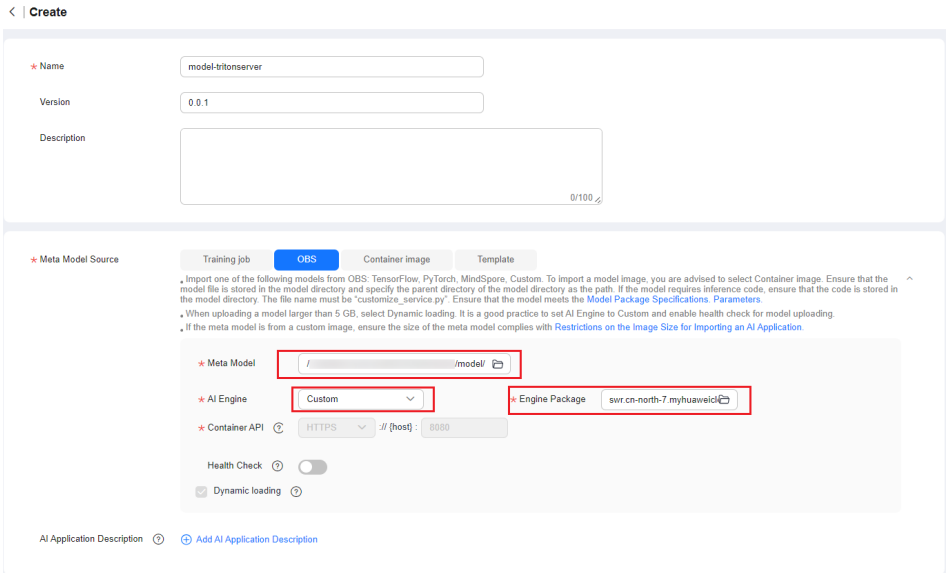
1. Cree un directorio **model** en OBS y cargue el archivo **triton_serving.sh** y la carpeta **llama_7b** en el directorio **model**.

Figura 5-23 Carga de archivos en el directorio **model**



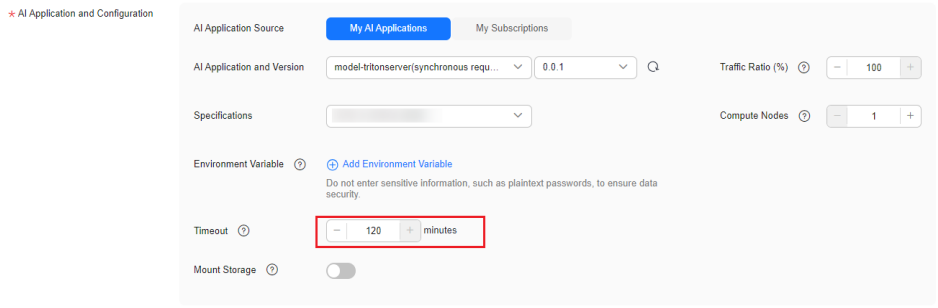
2. Cree una aplicación de IA. Configure **Meta Model Source** como **OBS** y seleccione el metamodelo desde el directorio **model**. Configure **AI Engine** como **Custom**. Establezca **Engine Package** en la imagen creada en **Paso 3**.

Figura 5-24 Creación de una aplicación de IA



3. Utilice la aplicación de IA creada como un servicio en tiempo real. Por lo general, el tiempo necesario para cargar e iniciar un modelo grande es largo al que el de un modelo común. Establezca **Timeout** en un valor adecuado. De lo contrario, el tiempo de espera puede transcurrir antes de la finalización de la puesta en marcha del modelo, y el despliegue puede fallar.

Figura 5-25 Despliegue de un servicio en tiempo real



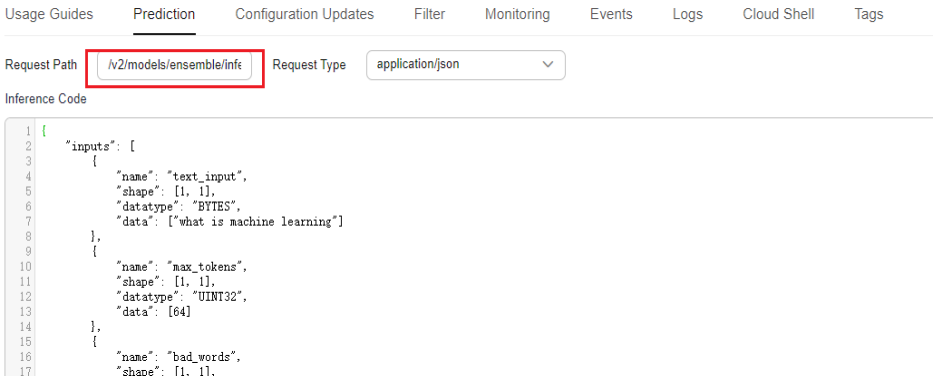
4. Llama al servicio en tiempo real para la inferencia del modelo de la fundación. Establezca la ruta de la solicitud en **/v2/models/ensemble/infer**. A continuación se muestra un ejemplo de llamada:

```
{
    "inputs": [
        {
            "name": "text_input",
            "shape": [1, 1],
            "datatype": "BYTES",
            "data": ["what is machine learning"]
        },
        {
            "name": "max_tokens",
            "shape": [1, 1],
            "datatype": "UINT32",
            "data": [64]
        },
        {
            "name": "bad_words",
            "shape": [1, 1],
            "datatype": "BYTES",
            "data": [""]
        },
        {
            "name": "stop_words",
            "shape": [1, 1],
            "datatype": "BYTES",
            "data": [""]
        },
        {
            "name": "pad_id",
            "shape": [1, 1],
            "datatype": "UINT32",
            "data": [2]
        },
        {
            "name": "end_id",
            "shape": [1, 1],
            "datatype": "UINT32",
            "data": [2]
        }
    ],
    "outputs": [
        {
            "name": "text_output"
        }
    ]
}
```


NOTA

- En "inputs", el elemento con el "name" "text_input" representa la entrada, y su campo "data" especifica una sentencia de entrada concreta. En este ejemplo, la sentencia de entrada es "what is machine learning".
- El elemento con el "name" "max_tokens" indica el número máximo de tokens de salida. En este caso, el valor es **64**.

Figura 5-26 Invocación a un servicio en tiempo real



----Fin

5.7 Acceso de alta velocidad a servicios de inferencia por interconexión de las VPC

Contexto

Al acceder a un servicio en tiempo real, es posible que necesite:

- Alto throughput y baja latencia
- Solicitudes de TCP o de RPC

Para cumplir con estos requisitos, ModelArts permite el acceso de alta velocidad por la interconexión de las VPC.

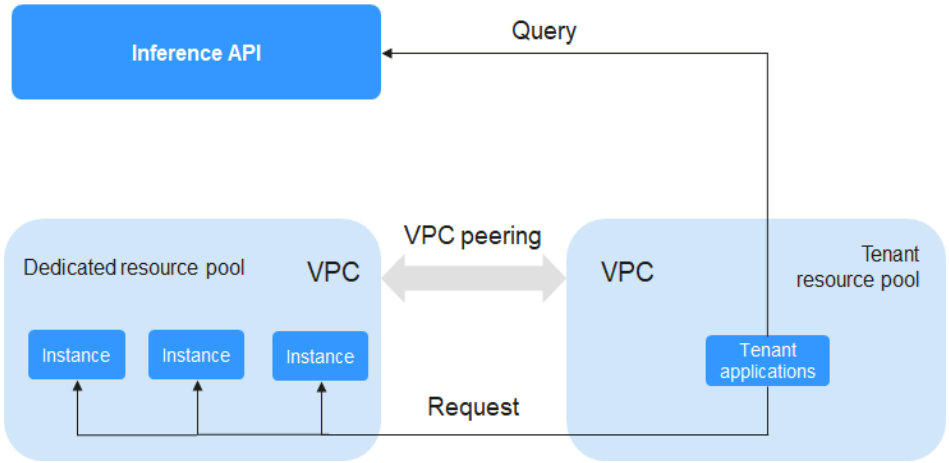
En el acceso de alta velocidad por la interconexión de las VPC, sus solicitudes de servicio se envían directamente a las instancias por la interconexión de las VPC, pero no con la plataforma de inferencia. Esto acelera el acceso al servicio.

NOTA

Las siguientes funciones disponibles con la plataforma de inferencia no estarán disponibles si utiliza el acceso de alta velocidad:

- Autenticación
- Distribución del tráfico por configuración
- Equilibrio de carga
- Alarma, monitoreo y estadísticas

Figura 5-27 Acceso de alta velocidad con interconexión de las VPC



Preparaciones

Despliegue un servicio en tiempo real en un grupo de recursos dedicado y asegúrese de que el servicio se está ejecutando.

AVISO

- Para obtener más detalles sobre cómo desplegar servicios en grupos de recursos dedicados de nueva versión, consulte **Actualizaciones amplias de funciones de gestión de grupos de recursos de ModelArts**.
- Solo los servicios desplegados en un grupo de recursos dedicado soportan acceso de alta velocidad por la interconexión de las VPC.
- El acceso de alta velocidad por interconexión de las VPC solo está disponible para los servicios en tiempo real.
- Debido al control del tráfico, existe un límite en la frecuencia con la que se puede obtener la dirección IP y el número de puerto de un servicio en tiempo real. La cantidad de llamadas de cada cuenta de tenant no puede superar las 2000 por minuto y la de cada cuenta de usuario de IAM no puede superar las 20 por minuto.
- El acceso de alta velocidad por interconexión de VPC solo está disponible para los servicios desplegados con las aplicaciones de IA importadas desde las imágenes personalizadas.

Procedimiento

Para habilitar el acceso de alta velocidad a un servicio en tiempo real por interconexión de las VPC, realice las siguientes operaciones:

1. **Interconecte el grupo de recursos dedicados a la VPC.**
2. **Cree un ECS en la VPC.**
3. **Obtenga la dirección IP y el número de puerto del servicio en tiempo real.**
4. **Acceda al servicio con la dirección IP y el número de puerto.**

Paso 1 Interconecte el grupo de recursos dedicados a la VPC.

Inicie sesión en la consola de gestión de ModelArts, seleccione **Dedicated Resource Pools** > **Elastic Cluster**, localice el grupo de recursos dedicado utilizado para el despliegue de servicio y haga clic en su nombre/ID para ir a la página de detalles del grupo de recursos. Obtenga la configuración de red. Vuelva a la lista del grupo de recursos dedicados, haga clic en la pestaña **Network**, localice la red asociada al grupo de recursos dedicados e interconéctela con la VPC. Después de acceder a la VPC, esta VPC aparecerá en las páginas de detalles de lista de redes y del grupo de recursos. Haga clic en la VPC para ir a la página de detalles.

Figura 5-28 Localización del grupo de recursos dedicados de destino

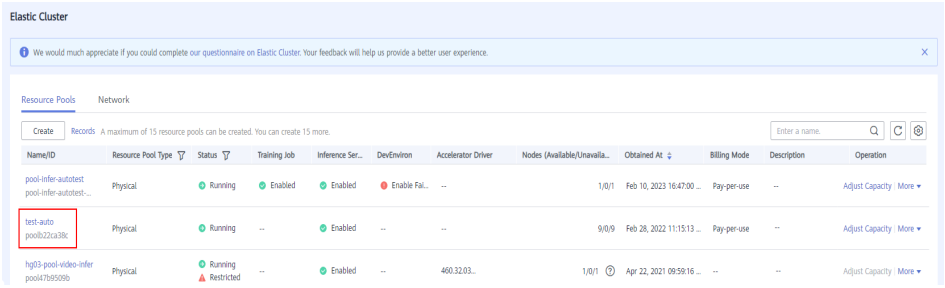


Figura 5-29 Obtención de la configuración de red

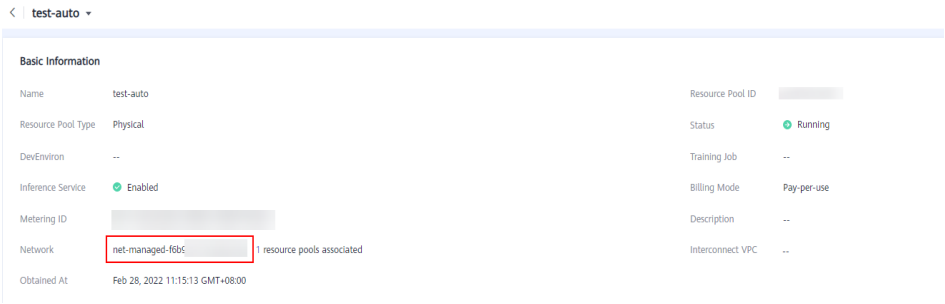
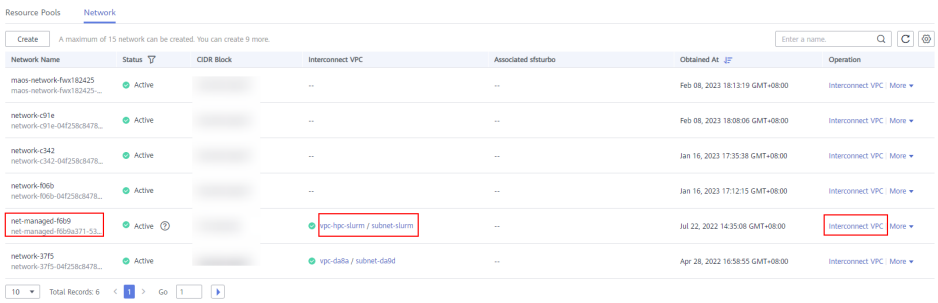


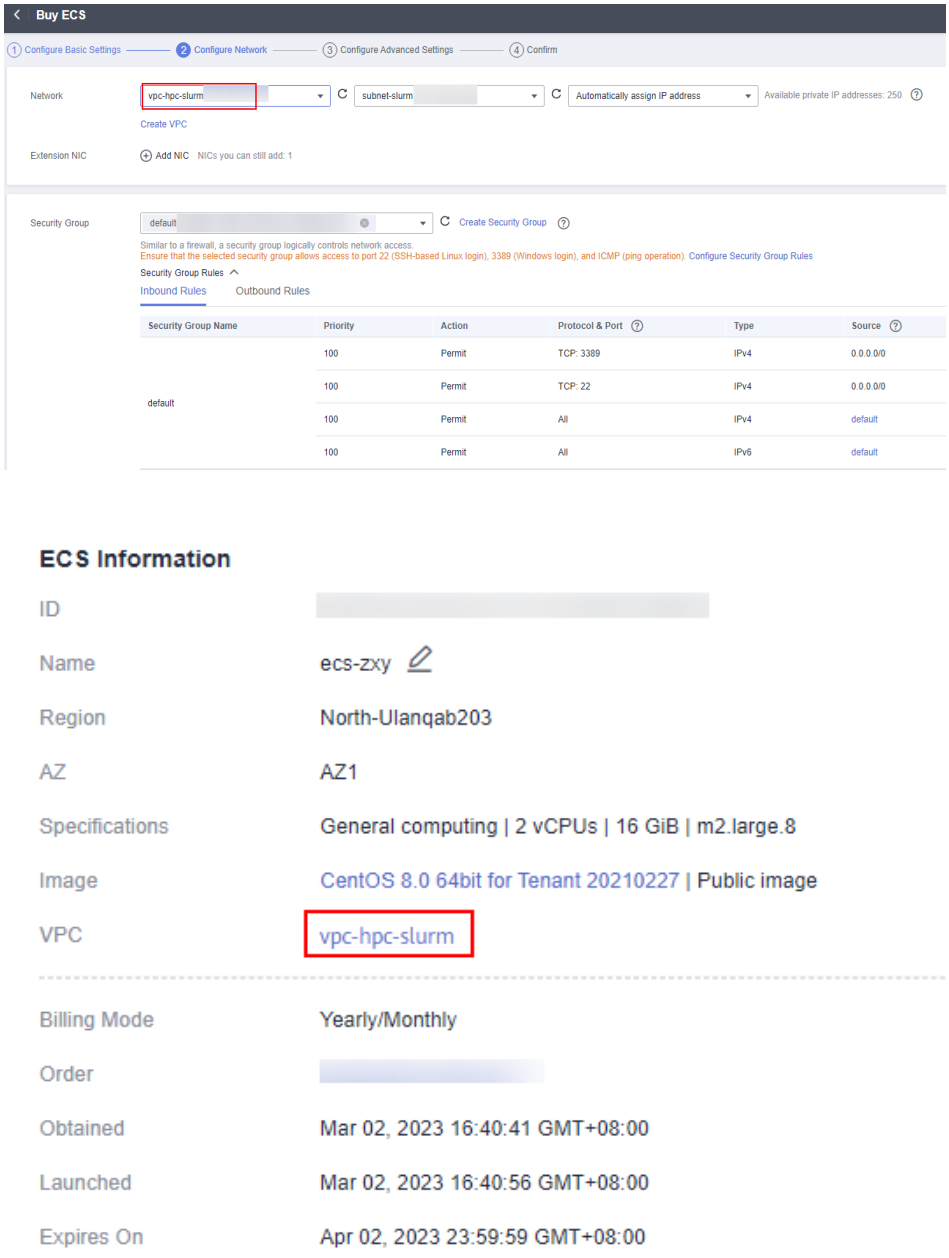
Figura 5-30 Interconexión de las VPC



Paso 2 Cree un ECS en la VPC.

Inicie sesión en la consola de gestión de ECS y haga clic en **Buy ECS** en la esquina superior derecha. En la página **Buy ECS**, configure los ajustes básicos y haga clic en **Next: Configure Network**. En la página **Configure Network**, seleccione la VPC conectada en **Paso 1**, configure otros parámetros, confirme los ajustes y haga clic en **Submit**. Cuando el estado del ECS cambia a **Running**, se crea el ECS. Haga clic en su nombre o ID para ir a la página de detalles del servidor y ver la configuración de VPC.

Figura 5-31 Selección de una VPC al comprar un ECS



Paso 3 Obtenga la dirección IP y el número de puerto del servicio en tiempo real.

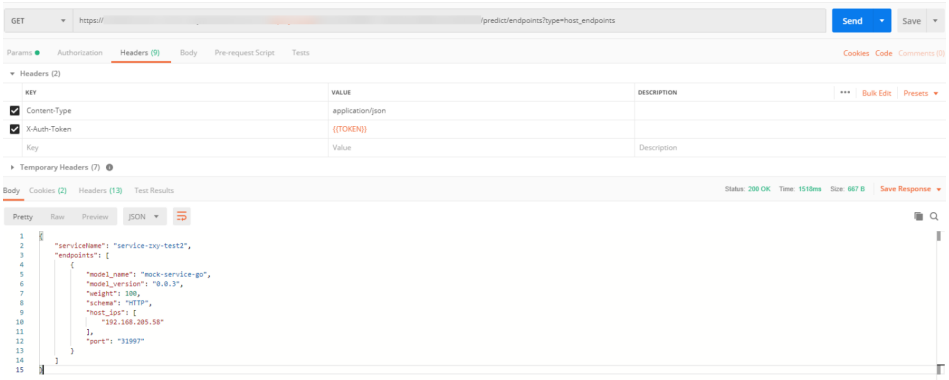
Se puede utilizar el software de GUI, por ejemplo, con Postman puede obtener la dirección IP y el número de puerto. De forma alternativa, inicie sesión en el ECS, cree un entorno de Python y ejecute el código para obtener la dirección IP del servicio y el número de puerto.

API:

```
GET /v1/{project_id}/services/{service_id}/predict/endpoints?type=host_endpoints
```

- Método 1: Obtenga la dirección IP y el número de puerto utilizando el software de GUI.

Figura 5-32 Ejemplo de la respuesta



- Método 2: Obtenga la dirección IP y el número de puerto usando Python.

  Es necesario modificar los siguientes parámetros en el código Python que aparece a continuación:

  – **project_id**: Su ID de proyecto. Para obtenerlo, consulte **Obtención de un ID y nombre de proyecto**.

  – **service_id**: ID del servicio, que puede consultarse en la página de detalles del servicio.

  – **REGION_ENDPOINT**: servicio de punto de conexión. Para obtenerlo, consulte **Punto de conexión**.

```
def get_app_info(project_id, service_id):
    list_host_endpoints_url = "{}/v1/{}/services/{}/predict/endpoints?
type=host_endpoints"
    url = list_host_endpoints_url.format(REGION_ENDPOINT, project_id,
service_id)
    headers = {'X-Auth-Token': X_Auth_Token}
    response = requests.get(url, headers=headers)
    print(response.content)
```

Paso 4 Acceda al servicio con la dirección IP y el número de puerto.

Inicie sesión en el ECS y acceda al servicio en tiempo real ejecutando comandos de Linux o creando un entorno de Python y ejecutando código de Python. Obtenga los valores de **schema**, **ip** y **port** de **Paso 3**.

- Ejecute el siguiente comando para acceder al servicio en tiempo real:

```
curl --location --request POST 'http://192.168.205.58:31997' \
--header 'Content-Type:  application/json' \
--data-raw '{"a":"a"}'
```

Figura 5-33 Acceso a un servicio en tiempo real

- Cree un entorno de Python y ejecute el código de Python para acceder al servicio en tiempo real.

```
def vpc_infer(schema, ip, port, body):
    infer_url = "{}://{}:{}"
    url = infer_url.format(schema, ip, port)
    response = requests.post(url, data=body)
    print(response.content)
```


NOTA

El acceso de alta velocidad no admite el balanceo de carga. Debe personalizar las políticas de equilibrio de carga cuando desplegar varias instancias.

----Fin

5.8 Desarrollo de procesos completos de servicios de WebSocket en tiempo real

Contexto

WebSocket es un protocolo de transmisión de red que admite comunicación dúplex completa a través de una sola conexión de TCP. Se sitúa en la capa de aplicación en un modelo de OSI. El protocolo de comunicación de WebSocket fue establecido por el IETF en 2011 como estándar RFC 6455 y complementado por RFC 7936. La API de WebSocket en Web IDL está estandarizada por W3C.

WebSocket simplifica el intercambio de datos entre el cliente y el servidor y permite que el servidor envíe datos de forma proactiva al cliente. En la API de WebSocket, si el handshake inicial entre el cliente y el servidor tiene éxito, se establecerá una conexión persistente entre ellos y los datos podrán transferirse bidireccionalmente.

Requisitos previos

- Tiene experiencia en el desarrollo de Java y está familiarizado con el empaquetado de JAR.
- Tiene conocimientos básicos y métodos de invocación de WebSocket.
- Está familiarizado con el método de creación de una imagen con Docker.

Restricciones

- WebSocket solo soporta el despliegue de servicios en tiempo real.
- WebSocket solo admite servicios en tiempo real desplegados mediante aplicaciones de IA importadas a partir de imágenes personalizadas.

Preparaciones

Antes de utilizar WebSocket de ModelArts para la inferencia, traiga su propia imagen personalizada. La imagen personalizada debe ser capaz de proporcionar servicios de WebSocket completos en un entorno autónomo, por ejemplo, completar handshakes de WebSocket e intercambiar datos entre el cliente y el servidor. La inferencia del modelo se implementa en la imagen personalizada, lo que incluye descargar el modelo, cargar el modelo, realizar el preprocesamiento, completar la inferencia y ensamblar el cuerpo de respuesta.

Procedimiento

Para desarrollar un servicio de WebSocket en tiempo real, realice las siguientes operaciones:

- **Carga de la imagen en SWR**
- **Creación de una aplicación de IA con la imagen**

- **Despliegue de la aplicación de IA como servicio en tiempo real**
- **Invocación al servicio de WebSocket en tiempo real**

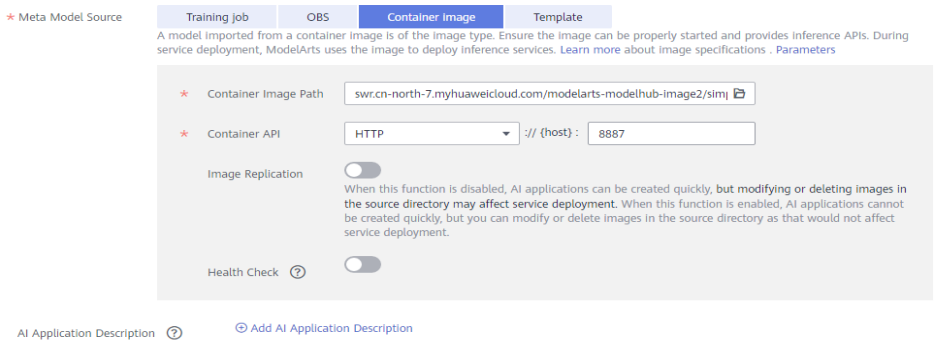
Carga de la imagen en SWR

Cargue la imagen local en SWR. Para obtener más detalles, consulte **¿Cómo puedo iniciar sesión en SWR y cargar imágenes en él?**

Creación de una aplicación de IA con la imagen

1. Inicie sesión en la consola de gestión de ModelArts, seleccione **AI Application Management** > **AI Applications** y haga clic en **Create** bajo **My AI Applications**. Aparece la página para crear una aplicación de IA.
2. Configure la aplicación de IA.
   - **Meta Model Source**: Seleccione **Container image**.
   - **Container Image Path**: seleccione la ruta especificada en **Carga de la imagen en SWR**.
   - **Container API**: configure este parámetro en función de los requisitos del sitio.
   - **Health Check**: conserve la configuración predeterminada. Si se ha configurado la comprobación de estado en la imagen, configure los parámetros de comprobación de estado en función de los configurados en la imagen.

   Figura 5-34 Parámetros de aplicación de IA

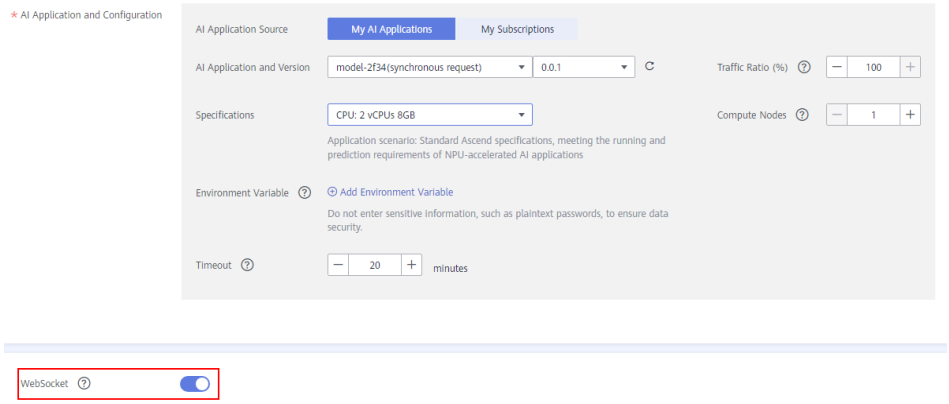


3. Haga clic en **Create now**. En la lista de aplicaciones de IA que aparece en pantalla, compruebe el estado de la aplicación de IA. Cuando cambia a **Normal**, se crea la aplicación de IA.

Despliegue de la aplicación de IA como servicio en tiempo real

1. Inicie sesión en la consola de gestión de ModelArts, seleccione **Service Deployment** > **Real-Time Services** y haga clic en **Deploy**.
2. Configure el servicio.
   - **AI Application and Version**: seleccione la aplicación de IA y la versión creadas en **Creación de una aplicación de IA con la imagen**.
   - **WebSocket**: Habilite esta función.

Figura 5-35 WebSocket



3. Haga clic en **Next**, confirme la configuración y luego haga clic en **Submit**. En la lista de servicios en tiempo real a la que se le redirigirá, compruebe el estado del servicio. Cuando cambia a **Running**, se despliega el servicio en tiempo real.

Invocación a un servicio en tiempo real de WebSocket

WebSocket no requiere la autenticación adicional. ModelArts WebSocket es compatible con WebSocket Secure, independientemente de si WebSocket o WebSocket Secure está activado en la imagen personalizada. WebSocket Secure solo admite la autenticación unidireccional, desde el cliente hasta el servidor.

Puede utilizar uno de los siguientes métodos de autenticación proporcionados por ModelArts:

- **Acceso autenticado mediante un token**
- **Acceso autenticado mediante una AK/SK**
- **Acceso autenticado con una aplicación**

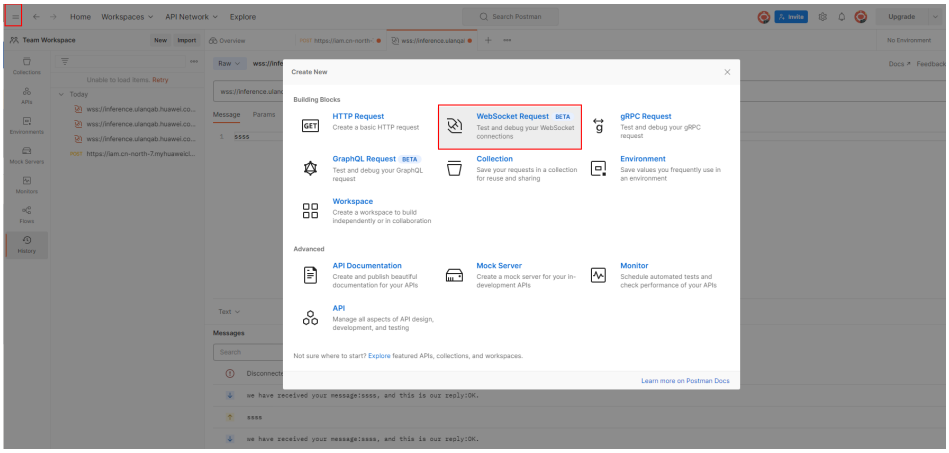
La siguiente sección utiliza el software de GUI Postman para predicción y la autenticación de token como ejemplo para describir cómo invocar a WebSocket.

1. **Establezca una conexión de WebSocket.**
2. **Intercambie datos entre el cliente de WebSocket y el servidor.**

Paso 1 Establecer una conexión de WebSocket.

1. Abra Postman de una versión posterior a 8.5, por ejemplo, 10.12.0. Haga clic en  en la esquina superior izquierda y elija **File** > **New**. En el cuadro de diálogo mostrado, seleccione **WebSocket Request** (versión de beta actual).

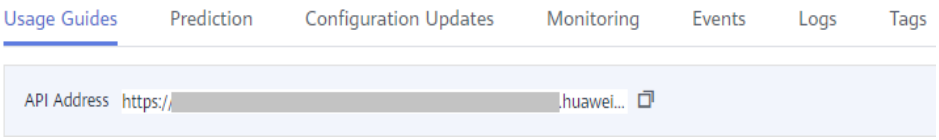
Figura 5-36 Solicitud de WebSocket



2. Configure los parámetros para la conexión de WebSocket.

   Seleccione **Raw** en la esquina superior izquierda. No seleccione **Socket.IO** (un tipo de implementación de WebSocket que requiere que tanto el cliente como el servidor funcionen con **Socket.IO**). En el cuadro de direcciones, introduzca la **API Address** obtenida en la ficha **Usage Guides** de la página de detalles del servicio. Si hay una dirección URL detallada en la imagen personalizada, agréguela al final de la dirección. Si **queryString** está disponible, agregue este parámetro a la columna **params**. Agregue información de autenticación al encabezado. El encabezado varía según el modo de autenticación, que es el mismo que el del servicio de inferencia compatible con HTTPS. Haga clic en **Connect** en la esquina superior derecha para establecer una conexión de WebSocket.

Figura 5-37 Obtención de la dirección API

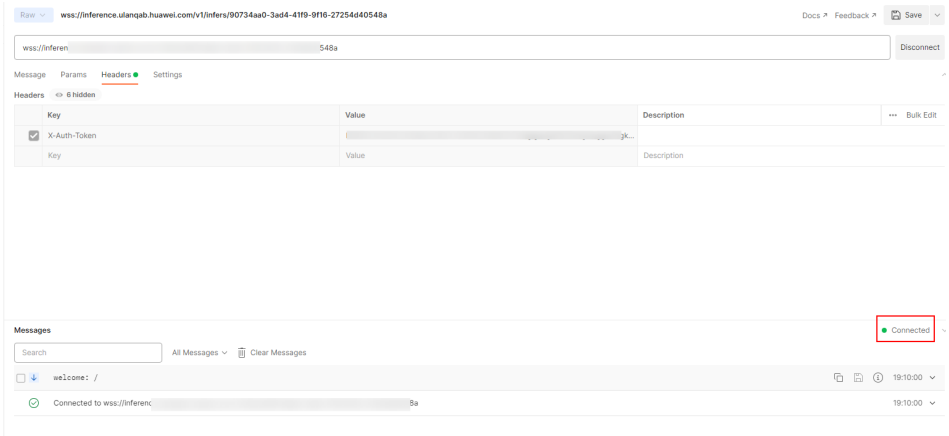


NOTA

- Si la información es correcta, aparecerá **CONNECTED** en la esquina inferior derecha.
- Si no se puede establecer la conexión y el código de estado es 401, verifique la autenticación.
- Si aparece una palabra clave como **WRONG_VERSION_NUMBER**, verifique si el puerto configurado en la imagen personalizada es el mismo que el configurado en WebSocket o WebSocket Secure.

A continuación se muestra una conexión de WebSocket establecida.

Figura 5-38 Conexión establecida



AVISO

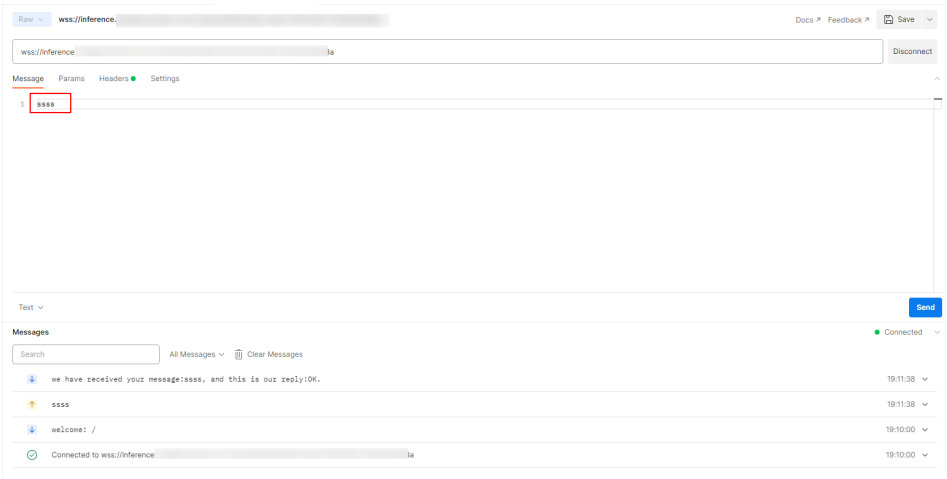
Verifique preferentemente el servicio de WebSocket proporcionado por la imagen personalizada. El tipo de implementación de WebSocket varía según la herramienta utilizada. Los problemas posibles son los siguientes: se puede establecer una conexión de WebSocket pero no se puede mantener, o la conexión se interrumpe tras una solicitud y es necesario volver a conectarse. ModelArts solo garantiza que no afectará al estado de WebSocket en una imagen personalizada (la dirección de API y el modo de autenticación pueden cambiarse en ModelArts).

Paso 2 Intercambie datos entre el cliente de WebSocket y el servidor.

Una vez establecida la conexión, WebSocket utiliza TCP para la comunicación dúplex completa. El cliente de WebSocket envía datos al servidor. Los tipos de implementación varían dependiendo del cliente, y el paquete de lib también puede ser diferente para el mismo idioma. Aquí no se tienen en cuenta los distintos tipos de implementación.

El formato de los datos enviados por el cliente no está limitado por el protocolo. Postman admite datos de texto, JSON, XML, HTML y Binary. Tomemos el texto como ejemplo. Introduzca los datos de texto en el cuadro de texto y haga clic en **Send** a la derecha para enviar la solicitud al servidor. Si el texto es demasiado grande, Postman puede ser suspendido.

Figura 5-39 Envío de datos



----Fin

