

SoftWare Repository for Container

User Guide (Ankara Region)

Issue 01
Date 2024-12-02



Copyright © Huawei Technologies Co., Ltd. 2024. All rights reserved.

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

Trademarks and Permissions



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

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

Notice

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

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

Security Declaration

Vulnerability

Huawei's regulations on product vulnerability management are subject to the *Vul. Response Process*. For details about this process, visit the following web page:

<https://www.huawei.com/en/psirt/vul-response-process>

For vulnerability information, enterprise customers can visit the following web page:

<https://securitybulletin.huawei.com/enterprise/en/security-advisory>

Contents

1 Service Overview	1
1.1 Introduction	1
1.2 Advantages	2
1.3 Application Scenarios	2
1.4 Basic Concepts	3
1.5 Notes and Constraints	4
1.6 Related Services	5
2 Overview	6
3 Permissions Management	7
3.1 Creating a User and Granting Permissions	7
4 Basics of Docker	9
5 Image Management	12
5.1 Pushing an Image Through a Container Engine Client	12
5.2 Obtaining a Long-Term Valid Docker Login Command	14
5.3 Obtaining a Long-Term Valid containerd Pull/Push Command	16
5.4 Uploading an Image Through the SWR Console	17
5.5 Pulling an Image	17
5.6 Setting Image Attributes	19
5.7 Sharing Private Images	20
5.8 Adding a Trigger	21
5.9 Adding an Image Retention Policy	23
6 Organization Management	25
7 User Permissions	28
8 FAQs	31
8.1 General FAQs	31
8.1.1 What Is SWR?	31
8.1.2 About SWR	31
8.1.3 How Do I Create a Container Image?	32
8.1.4 How Do I Create an Image Package?	36
8.1.5 Are There Quotas for SWR Resources?	37

8.1.6 Why Does Organization Creation Fail?.....	37
8.2 Login Issues.....	37
8.2.1 What Are the Differences Between Long-Term Valid Login Commands and Temporary Login Commands?.....	37
8.2.2 What Do I Do If My Login Command Expired?.....	38
8.3 Pushing an Image.....	38
8.4 Pulling an Image.....	39
8.5 Image Management FAQs.....	40
8.5.1 Can I Pull Images from SWR to a Local PC?.....	40
8.6 Troubleshooting.....	40
8.6.1 Why Does the Login Command Fail to Be Executed?.....	40
8.6.2 Why Does an Image Fail to Be Pushed Through a Container Engine Client?.....	42
8.6.3 Why Does an Image Fail to Be Uploaded Through SWR Console?.....	43
8.6.4 Why Does the docker pull Command Fail to Be Executed?.....	45
8.6.5 What Do I Do If an Error Occurs When I Call an API?.....	46
8.7 Other FAQs.....	46
8.7.1 Why Does a CCE Workload Cannot Pull an Image from SWR and the Message Indicating "Not Logged In" Is Displayed?.....	46
8.7.2 Why Is an Image Pushed Using a Container Engine Client to SWR Different in Size From One Uploaded Through the SWR Console?.....	47
8.7.3 How Many Tenants Can I Share an SWR Private Image With?.....	47

1 Service Overview

1.1 Introduction

SoftWare Repository for Container (SWR) allows you to easily manage the full lifecycle of container images and facilitates secure deployment of images for your applications.

SWR can either work with CCE or be used as an independent container image repository.

Features

- **Full lifecycle management of images**
SWR manages the whole lifecycle of your container images, including push, pull, and deletion.
- **Private image repository and access control**
Private image repository and fine-grained permission management allow you to grant different access permissions, namely, read, write, and edit, to different users.
- **Large scale image pull acceleration**
SWR uses the image pull acceleration technology to ensure faster image pull for CCE clusters in high concurrency scenarios.
- **Automatic deployment update through triggers**
Image deployment can be triggered automatically upon image update. Simply set a trigger to the desired image. Every time the image is updated, the application deployed with this image will be automatically updated.

Accessing SWR

The cloud platform provides a web-based management console and HTTPS-based APIs through which you can access the SWR service.

- **Using APIs**
If you want to integrate SWR into a third-party system for secondary development, use APIs to access SWR. For details, see *SWR API Reference*.

- Using the management console
Use this mode if you do not want to integrate SWR into a third-party system.

1.2 Advantages

Ease of Use

- You can directly push and pull container images without platform build or O&M.
- SWR provides an easy-to-use management console for full lifecycle management over container images.

Security and Reliability

- SWR uses HTTPS to secure image transmission, and provides multiple isolation mechanisms between and inside accounts to control access to images.
- Based on professional storage services, SWR provides highly reliable storage service for your container images.

Image Acceleration

SWR uses the image pull acceleration technology to ensure faster image pull for CCE clusters in high concurrency scenarios.

1.3 Application Scenarios

Image Lifecycle Management

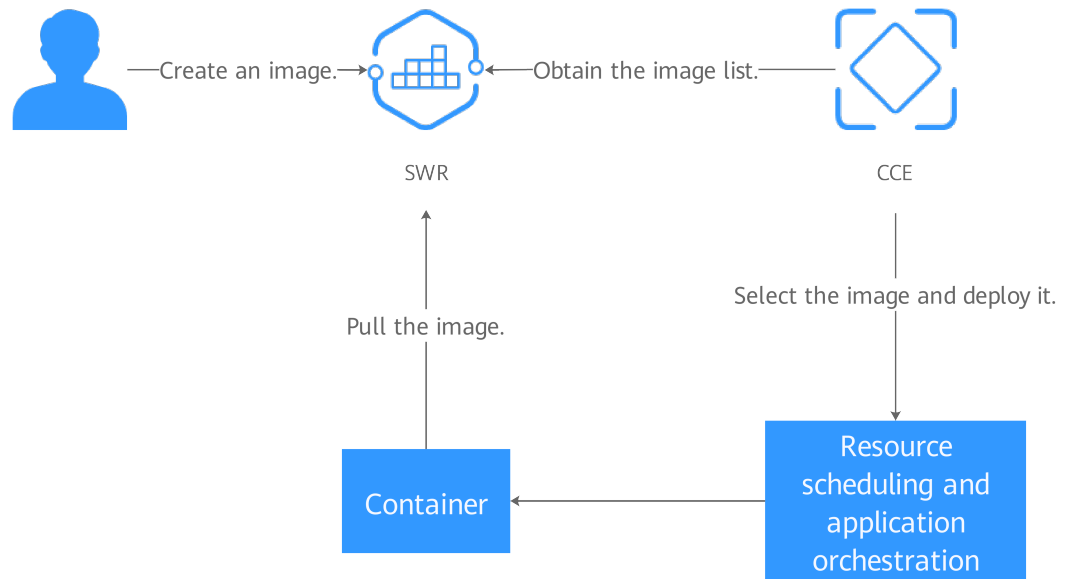
You can use SWR to build, push, pull, synchronize, and delete container images.

Advantages

- Pull acceleration ensures faster image pull for CCE clusters.
- Up to 99.999999999% image storage reliability is achieved by working with Object Storage Service (OBS).
- Fine-grained authorization allows you to control access to specific images and images in specific organizations.

Related service: Cloud Container Engine (CCE)

Figure 1-1 SWR working with CCE



1.4 Basic Concepts

Image

Images are like templates that include everything needed to run applications. When deploying containerized applications, you can use images from the Docker image center and your private image registries. For example, an image can contain a complete Ubuntu operating system, in which only the required programs and dependencies are installed. Docker images are used to create Docker containers. Docker provides an easy way to create and update your own images. You can also pull images created by other users.

Container

A container is a running instance of a Docker image. Multiple containers can run on one node. Containers are actually software processes. Unlike traditional software processes, containers have separate namespaces and do not run directly on a host.

Images become containers at runtime, that is, containers are created from images. Containers can be created, started, stopped, deleted, and suspended.

Repository

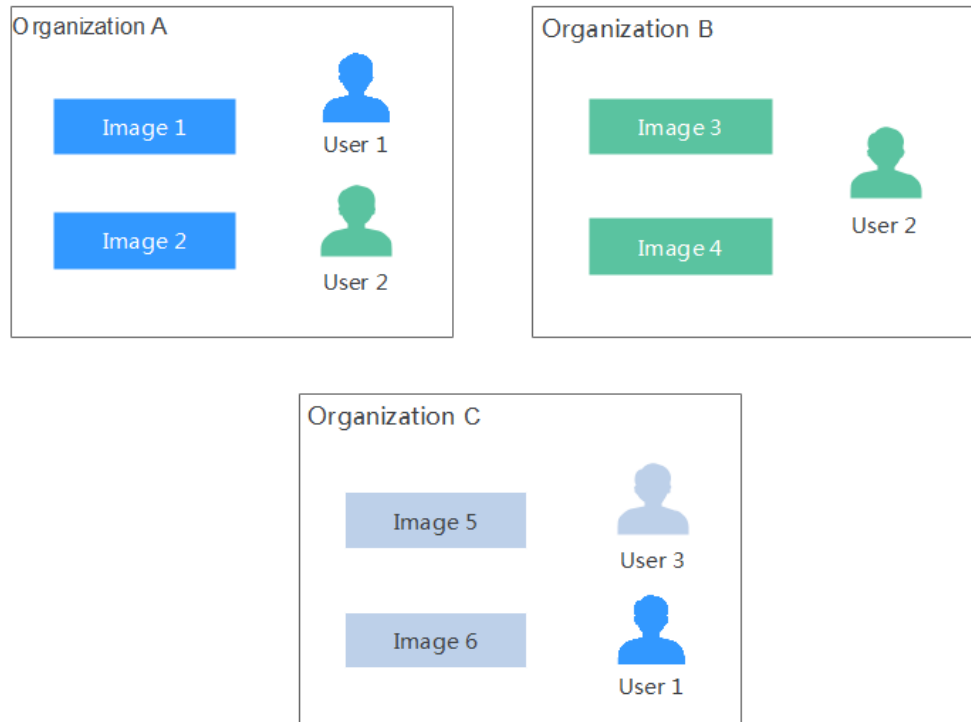
Image repositories are used for storing Docker images. An image repository hosts different versions of a specific containerized application.

Organization

Organizations are used to isolate image repositories. With each organization being limited to one company or department, images can be managed in a centralized and efficient manner. A user can access different organizations as long as the user

has corresponding permissions. Different permissions, namely read, write, and manage, can be assigned to different users in the same account.

Figure 1-2 Organization



1.5 Notes and Constraints

Quotas

Quotas are imposed on the number of organizations a user can create. [Table 1-1](#) lists the quotas imposed by SWR.

Table 1-1 SWR resource quotas

Resource Type	Quota
Organization	5

Requirements on Images to Upload

- If you use a container engine client to push images to SWR, the total number of image layers cannot exceed 20 at a time.
- If you use a container engine client to push images to SWR, each image layer cannot exceed 10 GB.

- If you use the SWR console to upload images, a maximum of 10 files can be uploaded at a time. The size of a single file (including the decompressed files) cannot exceed 2 GB.

1.6 Related Services

SWR works with other cloud services and requires permissions to access them.

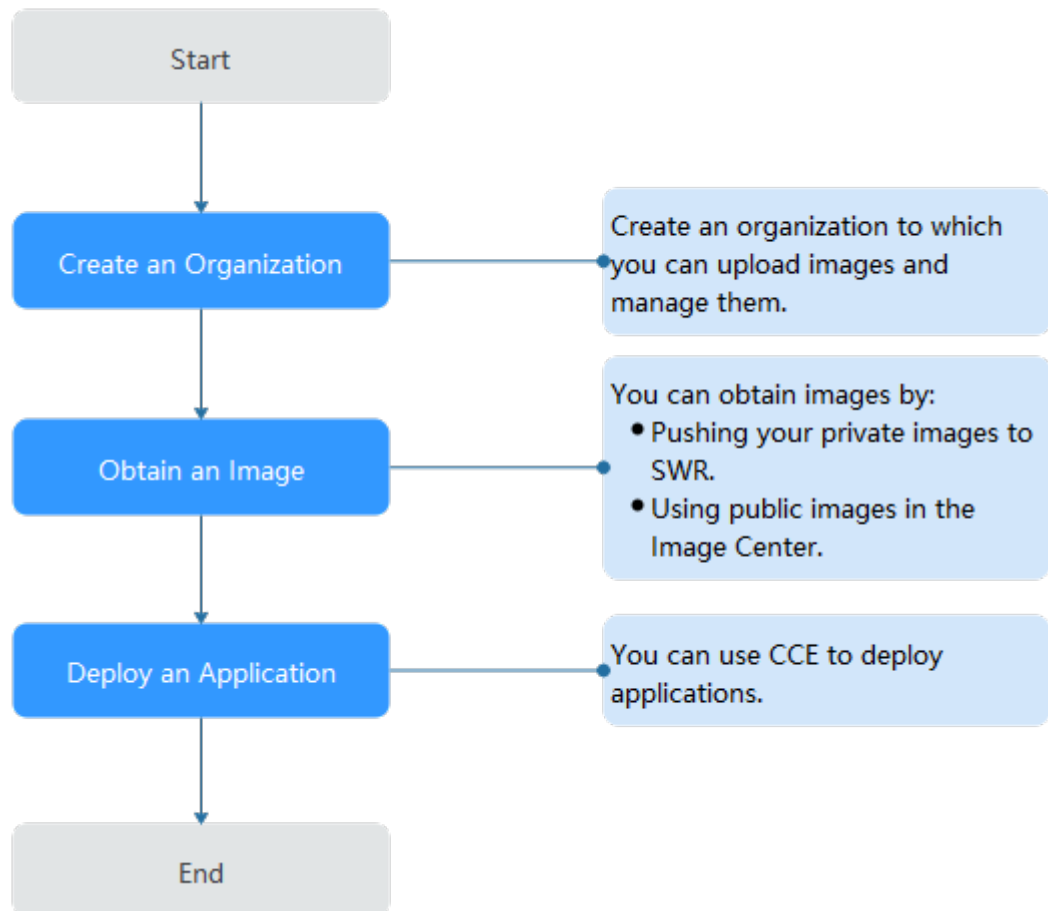
- **Cloud Container Engine (CCE)**
CCE is a high-performance, high-reliability service through which enterprises can manage containerized applications. CCE supports native Kubernetes applications and tools, allowing you to easily set up a container runtime environment on the cloud.
SWR works seamlessly with CCE to allow you to deploy your images held by SWR on CCE clusters.

2 Overview

SoftWare Repository for Container (SWR) allows you to easily manage the full lifecycle of container images and facilitates secure deployment of images for your applications.

SWR provides private image repositories and fine-grained permission management, allowing you to grant different access permissions, namely, read, write, and manage to different users. You can use triggers to automatically update applications when images are updated.

Figure 2-1 How SWR works



3 Permissions Management

3.1 Creating a User and Granting Permissions

This section describes how to use IAM for fine-grained permission management on your SWR resources. With IAM, you can:

- Create IAM users for employees based on your enterprise's organizational structure. Each IAM user will have their own security credentials for accessing SWR resources.
- Grant only the permissions required for users to perform a specific task.
- Entrust a cloud account or cloud service to perform efficient O&M on your SWR resources.

If your account does not need individual IAM users, you may skip this section.

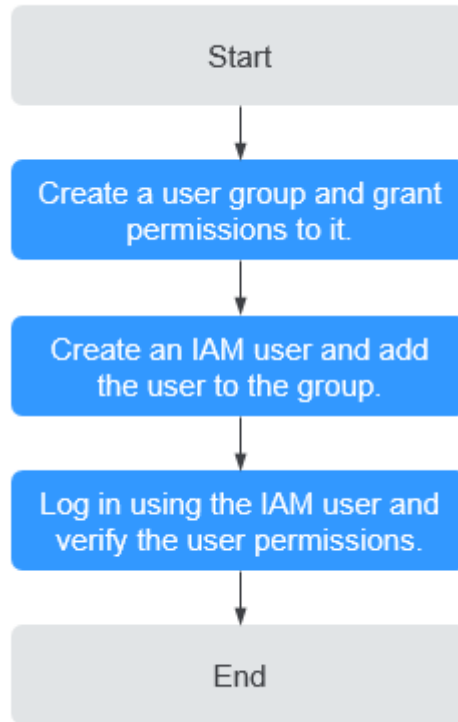
This section describes the procedure for granting permissions (see [Figure 3-1](#)).

Prerequisite

Learn about the permissions supported by SWR and choose policies or roles based on your requirements.

Process Flow

Figure 3-1 Process for granting SWR permissions



1. Create a user group and assign permissions.
Create a user group on the IAM console and click **Authorize** in the **Operation** column to assign the **SWR Administrator** permissions to the group.
2. Create an IAM user and add the user to a user group.
Create a user on the IAM console and add it to the user group created in **1** by choosing **Authorize** in the **Operation** column.
3. Log in as the IAM user and verify permissions.
Log in to the management console as the created user. Switch to the authorized region. Perform the following operations. If they can be successfully performed, the permissions are successfully granted.
 - a. Choose **Service List > Software Repository for Container**. The SWR console is displayed.
 - b. In the navigation pane on the left, choose **Organization Management**, click **Create Organization** in the upper right corner, and enter an organization name to create an organization.
 - c. In the navigation pane on the left, choose **My Images**, click **Upload Through SWR** in the upper right corner. Select the organization created in the previous step and a local image file. The image is successfully uploaded.

4 Basics of Docker

Docker is an open-source container engine which allows you to create a lightweight, portable, and self-sufficient container for any application. SWR is compatible with Docker, allowing you to use Docker CLI and APIs to manage your images.

Installing Docker

Before installing Docker, get a basic understanding of what Docker is and how it works. For more information, see [Docker Documentation](#).

Docker is compatible with almost all operating systems. Select a Docker version that best suits your needs. If you are not sure which Docker community edition to use, see <https://docs.docker.com/engine/install/>.

NOTE

- To use SWR, the Docker version must be between 1.11.2 (included) and 24.0.9 (included).
- Bind an elastic IP address first if your server runs in a private network as the installation requires Internet connection.

On a device running Linux, run the following commands to quickly install Docker:

```
curl -fsSL get.docker.com -o get-docker.sh
sh get-docker.sh
sudo systemctl daemon-reload
sudo systemctl restart docker
```

Building a Container Image

This section walks you through the steps of using a Dockerfile to build a container image for a simple web application. Dockerfile is a text file that contains all the instructions a user can call on the command line to build an image. A container image is a stack consisting of multiple layers. Each instruction creates a layer.

When using a browser to access a containerized application built from a Nginx image, you will see the default Nginx welcome page. In this section, you will build a new image based on the Nginx image to change the welcome message to **Hello, SWR!**

Step 1 Log in to the device running Docker as a root user.

Step 2 Run the following commands to create an empty file named **Dockerfile**:

```
mkdir mynginx
cd mynginx
touch Dockerfile
```

Step 3 Edit Dockerfile.

```
vim Dockerfile
```

Add the following instructions to the Dockerfile:

```
FROM nginx
RUN echo '<h1>Hello,SWR!</h1>' > /usr/share/nginx/html/index.html
```

In the preceding instructions:

- **FROM**: creates a layer from the base image. A valid Dockerfile must start with a **FROM** instruction. In this example, the **Nginx** image is used as the base image.
- **RUN**: executes a command to create a new layer. One of its syntax forms is **RUN <command>**. In this example, the **echo** command is executed to display **Hello, SWR!**

Save the changes and exit.

Step 4 Run **docker build [option] <context path>** to build an image.

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

- **-t nginx:v1**: specifies the image name and tag.
- **.**: indicates the path where the Dockerfile is located. All contents in this path are packed and sent to the Docker to build an image.

Step 5 Run the following command to check the created image. The command output shows that the nginx image has been created with a tag of v1.

```
docker images
```

```
----End
```

Creating an Image Package

This section describes how to compress a container image into a .tar or .tar.gz package.

Step 1 Log in to the device running Docker as a root user.

Step 2 Run the following command to list images.

```
docker images
```

Check the name and tag of the image to be compressed.

Step 3 Run the following command to compress the image into a package.

```
docker save [OPTIONS] IMAGE [IMAGE...]
```

 **NOTE**

OPTIONS: You can set this to **--output** or **-o**, indicating that the image is exported to a file.
The file should be in either **.tar** or **.tar.gz**.

Sample:

```
$ docker save nginx:latest > nginx.tar
$ ls -sh nginx.tar
108M nginx.tar

$ docker save php:5-apache > php.tar.gz
$ ls -sh php.tar.gz
372M php.tar.gz

$ docker save --output nginx.tar nginx
$ ls -sh nginx.tar
108M nginx.tar

$ docker save -o nginx-all.tar nginx
$ docker save -o nginx-latest.tar nginx:latest
```

----End

Importing an Image File

This section describes how to import an image package as an image using the **docker load** command.

There are two modes:

docker load < Path/File name.tar

docker load --input Path/File name.tar or **docker load -i Path/File name.tar**

Sample:

```
$ docker load --input fedora.tar
```


5 Image Management

5.1 Pushing an Image Through a Container Engine Client

Scenario

You can run **docker push** (Docker) or **ctr push** (containerd) on the server where the container engine client is installed to push an image to SWR.

Notes and Constraints

- Each image layer cannot exceed 10 GB.
- The Docker version must be between 1.11.2 (included) and 24.0.9 (included).


Prerequisites

You have created an organization in SWR. For details, see [Creating an Organization](#).

Docker

Take the **nginx:v1** image built in [Basics of Docker](#) as an example.

Step 1 Access SWR.

1. Log in to the SWR console and then the VM running Docker as the **root** user.
2. In the navigation pane on the left, choose **Dashboard** and click **Generate Login Command** in the upper right corner. On the displayed page, click  to copy the login command.

 **NOTE**

- A temporary login command is valid for 24 hours. For details about how to obtain a login command that will remain valid for a long term, see [Obtaining a Long-Term Valid Docker Login Command](#). After you obtain a long-term valid login command, your temporary login commands will still be valid as long as they are in their validity periods.
 - The domain name at the end of the login command is the image repository address. Record the address for later use.
3. Run the **docker login** command on your Docker client (a device that has Docker installed).

The message "Login Succeeded" will be displayed upon a successful login.

Step 2 Run the following command on the device where Docker is installed to label the **nginx** image:

```
docker tag [Image name 1:tag 1] [Image repository address] [Organization name] [Image name 2:tag 2]
```

In the preceding command:

- [Image name 1:tag 1]: Replace it with the actual name and tag of the image to be pushed.
- [Image repository address]: You can query the address on the SWR console, that is, the domain name at the end of the login command in [Step 1.2](#).
- [Organization name]: Replace it with the name of the organization created.
- [Image name 2: tag 2]: Replace it with the desired image name and tag.

Example:

```
docker tag nginx:v1 {Image repository address}/group/nginx:v1
```

Step 3 Push the image to the image repository by running the following command:

```
docker push [Image repository address] [Organization name] [Image name 2:tag 2]
```

Example:

```
docker push {Image repository address}/group/nginx:v1
```

The following information will be returned upon a successful push:

```
6d6b9812c8ae: Pushed
695da0025de6: Pushed
fe4c16cbf7a4: Pushed
v1: digest: sha256:eb7e3bbd8e3040efa71d9c2cacfa12a8e39c6b2ccd15eac12bdc49e0b66cee63 size: 948
```

To view the pushed image, refresh the **My Images** page.

----End

containerd

Step 1 Log in to the SWR console.

Step 2 In the navigation pane, choose **My Images** and click the name of your image.

Step 3 On the **Pull/Push** tab page, click **Generate upload instructions** and copy the image push command.

NOTE

The command is only valid for six hours after it is generated. To obtain a long-term valid command, see **Obtaining a Long-Term Valid containerd Pull/Push Command**.

Step 4 Log in to the VM running containerd as the **root** user.

Step 5 Run the command copied in **Step 3** (replace *{Tag}* with the new image tag).

```

[root@ ~]# docker push --rm --platform linux/amd64 localhost:5000/test:latest
WARNING: The 'mirrors' property of '[plugins.io.containerd.grpc.v1-registry]' is deprecated since containerd v1.6 and will be removed in containerd v2.0. Use 'config_path' instead.
WARNING: The 'configs' property of '[plugins.io.containerd.grpc.v1-registry]' is deprecated since containerd v1.5 and will be removed in containerd v2.0. Use 'config_path' instead.
sha256:569f5a31de11138255a708e15d6121d58963e0f8c4d99e83e37544f9d07 done
sha256:19dbdf6d3701a0212ca99ac076a21f777a08380edd50c160528066ef8e done
Pushed: 0.3 s
total: 1.7 Ki (5.5 KiB/s)
    
```

Step 6 Check whether the image is pushed successfully.

----End

5.2 Obtaining a Long-Term Valid Docker Login Command

Scenario

This section describes how to obtain a Docker login command that is permanently valid.

NOTE

For security purposes, you are advised to obtain the command in a development environment.

Process

You can obtain a long-term valid login command as the following process:


Figure 5-1 Process



Procedure

Step 1 Obtain the programmatic access permission. (If the current user has the permission, skip this step.)

1. Log in to the management console as an administrator.
2. Click in the upper left corner and select a region and a project.
3. Click in the navigation pane on the left and choose **Management & Deployment > Identity and Access Management**.

4. Enter the name of the user to whom you want to grant the programmatic access permission in the search box on the **Users** page.
5. Click the user to go to its details page.
6. Click  next to **Access Type**.
7. Select **Programmatic access**. (You can select only programmatic access or both access types.)

Step 2 Obtain the region, project name, and image repository address.

1. Log in to the management console, click your username in the upper right corner, and click **My Credentials**.
2. In the project list on the **API Credentials** page, search for the project corresponding to the current region.
3. Obtain the image repository address by referring to **Step 1.2**. The domain name at the end of the login command is the image repository address.

Step 3 Obtain an AK/SK.

 **NOTE**

The access key ID (AK) and secret access key (SK) are a pair of access keys used together to authenticate users who wish to make API requests. The AK/AS pair provides functions similar to a password. If you already have an AK/SK, skip this step.

1. Log in to the management console, click your username in the upper right corner, and click **My Credentials**.
2. In the navigation pane on the left, choose **Access Keys** and click **Create Access Key**.
3. Download the access key, which includes the AK and SK.

 **NOTE**

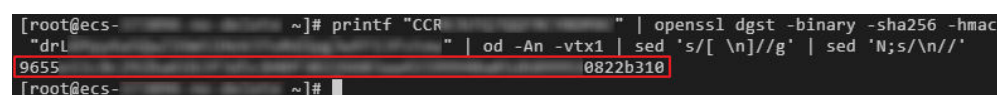
Keep the access key secure and do not disclose it to any unauthorized personnel.

Step 4 Log in to a Linux PC and run the following command to obtain the login key:

```
printf "$AK" | openssl dgst -binary -sha256 -hmac "$SK" | od -An -vtx1 | sed 's/[ \n]//g' | sed 'N;s/\n/'
```

In the command, *\$AK* and *\$SK* indicate the AK and SK obtained in **Step 3**.

Figure 5-2 Sample command output



```
[root@ecs- ~]# printf "CCR" | openssl dgst -binary -sha256 -hmac "drL" | od -An -vtx1 | sed 's/[ \n]//g' | sed 'N;s/\n/'
9655 0822b310
[root@ecs- ~]#
```

Step 5 Put the information you obtained in the following format to generate a long-term valid login command:

```
docker login -u [Regional project name]@[AK] -p [Login key] [Image repository address]
```

In the command, the regional project name and image repository address are obtained in **Step 2**, the AK in **Step 3**, and the login key in **Step 4**.

 NOTE

The login key is encrypted and cannot be decrypted. Therefore, other users cannot obtain the SK from -p.

The login command can be used on other devices.

Step 6 Run the **history -c** command to clear the operation records.

----End

5.3 Obtaining a Long-Term Valid containerd Pull/Push Command

Scenario

This section describes how to obtain a containerd pull/push command that is permanently valid.

 NOTE

- For security purposes, you are advised to obtain the commands in a development environment.
- Ensure that you have permission to access the IAM service.

Procedure

Step 1 Obtain the programmatic access permission by referring to [Step 1](#).

Step 2 Obtain the resource space name, image repository address, AK, and login key by referring to [Step 2](#) to [Step 4](#).

Step 3 Concatenate the obtained information to form a long-term valid containerd command.

1. Image pull command

```
ctr image pull --user [Resource space name] @[AK]: [Login key] [Image repository address]
```

In the command, the resource space name and image repository address are obtained in [Step 2](#), the AK in [Step 3](#), and the login key in [Step 4](#).

2. Image push command

```
ctr image push --user [Resource space name] @[AK]: [Login key] [Image repository address]
```

In the command, the resource space name and image repository address are obtained in [Step 2](#), the AK in [Step 3](#), and the login key in [Step 4](#).

 NOTE

- The login key is encrypted and cannot be decrypted into an SK.
- The commands can be executed on other containerd clients to pull and push images.

----End

5.4 Uploading an Image Through the SWR Console

Scenario

This section describes how to upload an image to SWR through the SWR console.

Notes and Constraints

- A maximum of 10 files can be uploaded at a time. The size of a single file (including the decompressed files) cannot exceed 2 GB.
- Only image file packages created by Docker 1.11.2 to 24.0.9 can be uploaded.

Prerequisite

- You have created an organization in SWR. For details, see [Creating an Organization](#).
- The image has been saved as a **.tar** or **.tar.gz** package. For details, see [Creating an Image Package](#).

Procedure

Step 1 Log in to the SWR console.

Step 2 In the navigation pane, choose **My Images**. Then click **Upload Through SWR**.

Step 3 On the page displayed, select an organization. Then, click **Select File** to upload the desired image file.

NOTE

If you select multiple images to upload, the system uploads them one by one. Concurrent upload is not supported.

Step 4 Click **Start Upload**.

When the upload progress is complete, the image is successfully uploaded.

----End

5.5 Pulling an Image

Scenario

You can use Docker or containerd to pull images from SWR.


Docker

Step 1 Log in to the VM running Docker as the **root** user.

Step 2 Obtain a login command by referring to [Step 1](#) and access SWR.

Step 3 Log in to the SWR console.

Step 4 In the navigation pane, choose **My Images** and click the target image.

Step 5 On the **Image Tags** tab page, in the same row as the target image tag, click  in the **Image Pull Command** column to copy the command.

Step 6 Run the **image pull** command obtained in **Step 5** on the VM.

Run the **docker images** command to check whether the images are successfully pulled.

```
# docker images
REPOSITORY          TAG          IMAGE ID       CREATED        SIZE
xxx/group/nginx     v2.0.0      22f2bf2e2b4f  5 hours ago   22.8MB
```

Step 7 (Optional) Run the following command to save the image as an archive file:

docker save [*Image name:tag name*] > [*Archive file name*]

----End

containerd

Step 1 Log in to the SWR console.

Step 2 In the navigation pane, choose **My Images** and click the name of your image.

Step 3 On the **Tags** tab page, click **containerd command** in the **Operation** column to copy the image pull command. Alternatively, go to the **Pull/Push** tab page to copy the image pull command.

NOTE

The command is only valid for six hours after it is generated. To obtain a long-term valid command, see [Obtaining a Long-Term Valid containerd Pull/Push Command](#).

Step 4 Log in to the VM running containerd as the **root** user.

Step 5 Run the command copied in **Step 3**.

- If the command was copied from the **Operation** column, run it as follows.

```
[root@h 9be186eac1d40ea2fb68f15282 sur. mjuaweicloud.com] ~/kritis-validation-admission:24.3.14_aarch64
Image is up to date for sha256:68d22ad99915e044af2a15ef0940c958f9884dc228924d26726e9ba63287b6cb
```

- If the command was copied from the **Pull/Push** tab page, run it as follows (replace *{Tag}* with the new image tag).

```
[root@h 518156cca1cf9fa9b0f6b24e748f6c sur. mjuaweicloud.com] ~/kritis-validation-admission:24.3.14_aarch64
WARN[0000] DEPRECATION: The `mirrors` property of `plugins.io.containerd.grpc.v1.cri`.registry is deprecated since container
d v1.5 and will be removed in containerd v2.0. Use `config_path` instead.
sur.cn-north-7.mjuaweicloud.com ~/kritis-validation-admission:24.3.14_aarch64: resolved |.....|
.....|
manifest-sha256:e34f8144b428fdaff2a9f82f2aa9d5291a04340a400ea166da9f9c9e91be2d32: done |.....|
.....|
config-sha256:68d22ad99915e044af2a15ef0940c958f9884dc228924d26726e9ba63287b6cb: done |.....|
.....|
layer-sha256:4ec0f98f626c28abeab08113f5994a423bc20b7e9b107e05a0c0f2a3d40e47be: done |.....|
.....|
layer-sha256:bc7f43b68355f37558f4e315f2b1b64d03412622cd66e794836db13324fb7ce: done |.....|
.....|
layer-sha256:825d167cd3557957e6c3ef139683e897102d4eefbd7c2886359e2f54e08cbd7e: done |.....|
.....|
layer-sha256:3f954aa41fcd54743e0d0947d4d03ec38635f7307410bbb98a1d2b63b6fe74ab: done |.....|
.....|
layer-sha256:51482c154bee556e8d31a697f806748de512aeecc822c0d4a88fe830d31a3782: done |.....|
.....|
elapsed: 6.1 s total: 372.7 (61.1 MiB/s)
.....|
unpacking linux/amd64 sha256:e34f8144b428fdaff2a9f82f2aa9d5291a04340a400ea166da9f9c9e91be2d32...
done: 1.293649364s
```

Step 6 Check whether the image is pulled successfully.

- If the command was copied from the **Operation** column, run **crictl images** to check whether the pull is successful.

```

root@ ~# ctrctl pull --creds   

9be186eac1d40ea2bfb68f15282 swr. .mjhuaawcloud.com/ /kritis-validation-admission:24.3.14_aarch64
Image is up to date for sha256:68d22ad99915e044af2a15ef0940c950f9804dc22924d26726e9ba6328766cb
root@ ~# ctrctl images
IMAGE TAG IMAGE ID SIZE
docker.io/library/cce-pause 3.1 c96088c71666e 687KB
swr. .mjhuaawcloud.com/ /kritis-validation-admission 24.3.14_aarch64 68d22ad99915e 394KB
swr. .mjhuaawcloud.com/huofficial/everest-csi-driver-init 2.4.61 f1e3147427fcf 129MB
swr. .mjhuaawcloud.com/huofficial/everest 2.4.61 24f98419db3af 186MB
swr. .mjhuaawcloud.com/op_svc_apm/icagent 5.31.32.41 b9f53a90a0d1b 219MB
root@ ~#

```

- If the command was copied from the **Pull/Push** tab page, run **ctr images list** to check whether the pull is successful.

```

root@ ~# ctr images list
WARNING[0000] DEPRECATION: The `mirrors` property of `plugins.io.containerd.grpc.v1.cri`.registry is deprecated since containerd v1.5 and will be removed in containerd v2.0. Use `config_path` instead.
REF DIGEST TYPE SIZE PLATFORMS LABELS
swr.cn-north-7.mjhuaawcloud.com/h30013402/kritis-validation-admission:24.3.14_aarch64 application/vnd.docker.distribution.manifest.v2+json sha256:e34f8144b428fdaff2a9f82f2aa945291a04340a400ea166da9f9cee91be2432 375.5 MiB linux/arm64 -

```

----End

5.6 Setting Image Attributes

Scenario

After uploading an image, you can set image attributes, including its type (public or private), category, and description.

Public images can be pulled by all users; whereas the access to private images requires corresponding permissions. You can add permissions, namely, read, write, and manage, to allow users to access your private images. For details, see [Granting Permissions for a Specific Image](#).

Procedure

- Step 1** Log in to the SWR console.
- Step 2** In the navigation pane, choose **My Images** and click the desired image.
- Step 3** On the details page, click **Edit** in the upper right corner. On the page displayed, set **Type (Public or Private)**, **Category**, and **Description**, and click **OK**.

Table 5-1 Editing an image

Parameter	Description
Organization	The organization to which the image belongs
Image	Image name

Parameter	Description
Type	<p>The following options are available:</p> <ul style="list-style-type: none"> • Public • Private <p>NOTE Public images can be pulled and used by all users.</p> <ul style="list-style-type: none"> • If your machine and the image repository are in the same region, you can access the image repository over private networks. • If your machine and the image repository are in different regions, the node must have access to public networks to pull images from the image repository.
Category	<p>The following options are available:</p> <ul style="list-style-type: none"> • Application server • Linux • Windows • Arm • Framework & Application • Database • Language • Others
Description	Image description. Enter a maximum of 30,000 characters.

----End

5.7 Sharing Private Images

Scenario

You can share your **private images** with other accounts and grant the accounts permissions to pull the images.

A user under the account with which you shared the image can then log in to the SWR console to view the image by clicking **My Images > Shared Images**. On the tab page, the user can click the target image to check its detailed information, including the image tag and image pull command.

Notes and Constraints

- Only private images can be shared. Public images cannot be shared.
- Only users authorized to manage the private images can share images. The users with whom you share your images only have the read-only permission, which only allows them to pull the images.
- You can share images only with accounts in the same region. Cross-region image sharing is not supported.

Procedure

- Step 1** Log in to the SWR console.
- Step 2** In the navigation pane, choose **My Images** and click the target image.
- Step 3** On the details page, click the **Sharing** tab.
- Step 4** Click **Share Image**. Set parameters based on [Table 5-2](#), and click **OK**.

Table 5-2 Sharing an image

Parameter	Description
Share With	Enter an account name with which you want to share the image.
Valid Until	Set a validity period. If you want the image to be permanently accessible to the account, select Permanently valid .
Description	Enter a maximum of 1,000 characters.
Permission	Only the Pull permission is supported currently.

- Step 5** To view all the images that you have shared, choose **My Images** in the navigation pane, click the **Private Images** tab, and select **Display only shared images**.

----End

5.8 Adding a Trigger

Scenario

SWR works with Cloud Container Engine (CCE) to achieve automatic application update. When images are updated, these new images can be automatically deployed to update the applications that use these images. You only need to add a trigger to the desired images. Every time these images are updated, they can trigger automatic updates of the applications that use them.

Prerequisite

A containerized application has been created on CCE by using an image from SWR.

To create an application, log in to the CCE console and create a workload.

Procedure

- Step 1** Log in to the SWR console.
- Step 2** In the navigation pane on the left, choose **My Images**, and click the target image.
- Step 3** Click the **Triggers** tab, then click **Add Trigger**. On the page displayed, configure the following parameters according to [Table 5-3](#) and click **OK**.

Table 5-3 Trigger

Parameter	Description
Name	The name can contain 1 to 64 characters, and must start with a letter. Only letters, digits, underscores (_), and hyphens (-) are allowed. The name cannot end with an underscore or hyphen. Consecutive underscores or hyphens are not allowed and an underscore cannot be placed next to a hyphen.
Condition	The following trigger conditions are supported: <ul style="list-style-type: none"> • All tags: Trigger when any image tags are generated or updated. • Specific tag: Trigger when a specific image tag is generated or updated. • Tags matching regular expression: Trigger when an image tag that matches the specified regular expression is generated or updated. The regular expression rules are as follows: <ul style="list-style-type: none"> - *: matches any field that does not contain the path separator /. - **: matches any field that contains the path separator /. - ?: Matches any single character except /. - {option 1, option 2, ...}: matches multiple options.
Operation	Operation that will be triggered when the conditions you set are met. Currently, only application update is supported. You need to specify the application to be updated and the container image of the application.
Status	Select Enable .
Trigger Type	Select CCE .
Application	Select the container whose image you want to update.

----End

Example

A Deployment named **nginx** is created using the Nginx v1 image. The Deployment provides service to external systems with a welcome page displaying **Hello, SWR!**

Figure 5-3 Nginx deployment

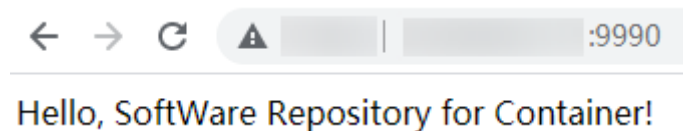


1. Add a trigger to the Nginx image.
Set **Name** to **All_tags**, **Condition** to **All_tags**, and select the application and all its containers that use the Nginx image.
2. Check the image tags. The Nginx image v3 is pushed to SWR. The welcome page of the Deployment created using this new image should display **Hello, SoftWare Repository for Container!**
3. Check whether the deployment is triggered successfully.

On the **Triggers** tab page, click  and the trigger is successful.

The welcome page of the Deployment displays **Hello, SoftWare Repository for Container!**

Figure 5-4 Updated Nginx



5.9 Adding an Image Retention Policy

Scenario

You can add a retention policy to an image in SWR to automatically delete any unused image tags. The policy takes effect immediately after you set it. There are two types of policies:

- Number of days: keeping only image tags that have been pushed to SWR within a certain number of days.
- Number of tags: keeping only a certain number of the most recent image tags.

You can configure filters for your retention policy to prevent certain image tags from being affected by the retention policy.

Notes and Constraints

Only one retention rule can be added to an image. If you want to add a new retention policy, you must delete the existing policy.

Procedure

- Step 1** Log in to the SWR console.
- Step 2** In the navigation pane on the left, choose **My Images**, and click the target image.
- Step 3** On the **Retention** tab page, click **+**. Configure the policy based on [Table 5-4](#) and click **OK**.

Table 5-4 Adding an image retention policy

Parameter	Description
Policy Type	There are two types of retention policies: <ul style="list-style-type: none"> • Number of days: keeping only image tags that have been pushed to SWR within a certain number of days. • Number of tags: keeping only a certain number of the most recent image tags.
Count Limit (Number of days)	When you set Policy Type to Number of days , the value of Count Limit indicates how many days an image tag can be stored. The value should be an integer ranging from 1 to 365.
Count Limit (Number of tags)	When you set Policy Type to Number of tags , the value of Count Limit indicates the maximum number of the most recent image tags to be retained. The value should be an integer ranging from 1 to 1,000.
Tag Filter	Enter image tags that you do not want this retention policy to apply to.
Regular Expression Filter	Enter a regular expression. Image tags meeting this regular expression will not be affected by this retention policy.

After the retention policy is added, SWR immediately applies the policy and displays deleted image tags (if any) in the **Retention Logs** area.

----End

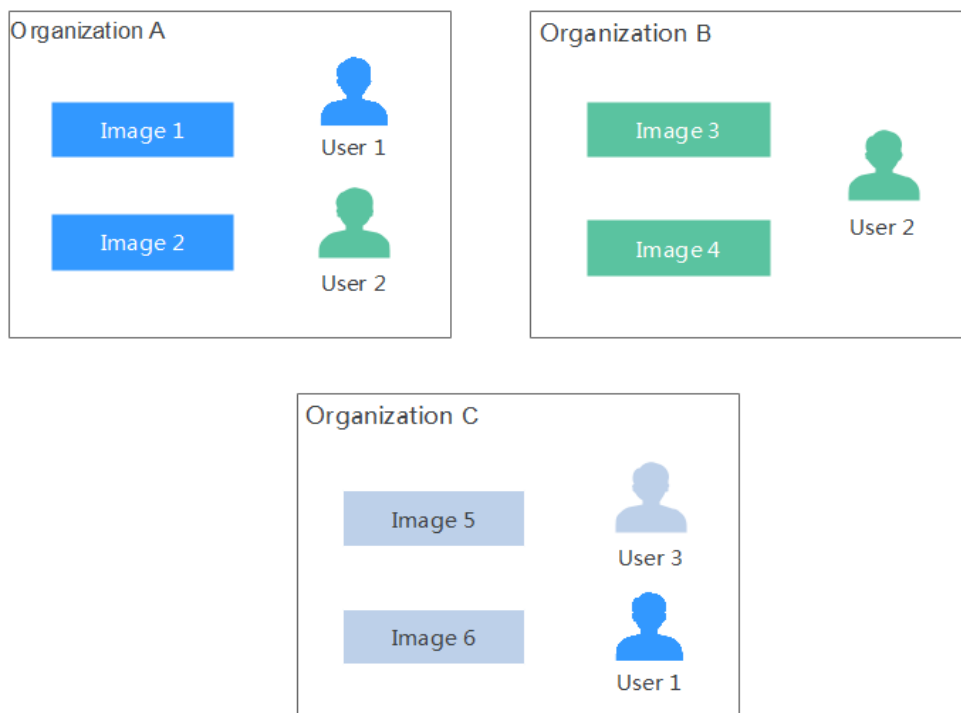
6 Organization Management

Scenario

Organizations enable efficient management of images. Organizations are used to isolate image repositories. With each organization being limited to one company or department, images can be managed in a centralized and efficient manner. An image name needs to be unique within an organization. The same user can access different organizations as long as the user has sufficient permissions, as shown in [Figure 6-1](#).

You can grant different permissions, namely, read, write, and manage, to users created by the same account. For details, see [User Permissions](#).

Figure 6-1 Organization



Creating an Organization

You can create organizations based on the organizational structure of your enterprise to facilitate image resource management. Create an organization before you push an image.

Step 1 Log in to the SWR console.

Step 2 In the navigation pane on the left, choose **Organization Management** and click **Create Organization**. On the page displayed, specify **Organization Name** and click **OK**.

NOTE

- The organization name must be globally unique. If a message is displayed indicating that the organization already exists, the organization name may have been used by another user. Use another organization name.
- After a tenant is deleted, residual organization resources may exist. In this case, the message indicating that the organization already exists could also be displayed when you create an organization. Use another organization name.

----End

Viewing the Images of an Organization

After you create an organization and push images to it, you can view the image list of the organization.

Step 1 Log in to the SWR console.

Step 2 In the navigation pane, choose **Organization Management**. On the page displayed, click the desired organization name in the list.

Step 3 To view the images of this organization, click the **Images** tab.

----End

Deleting an Organization

Before deleting an organization, delete all the images in the organization.

Step 1 Log in to the SWR console.

Step 2 In the navigation pane, choose **Organization Management**. On the page displayed, click the desired organization name in the list.

Step 3 Click **Delete** in the upper right corner and click **Yes**.

----End

NOTICE

Before you delete a tenant, delete its organizations first; otherwise, residual organization resources may exist. When you create an organization that has the same name with the residual organization, a message is displayed indicating that the organization already exists.

7 User Permissions

Scenario

To manage SWR permissions, you can use Identity and Access Management (IAM). If you have the SWR Administrator or Tenant Administrator permission, you become an admin user of SWR accounts. You can grant permissions to other IAM users in SWR.

If you are not an SWR account admin user, you can request an SWR account admin user to grant you permissions to read, write, or manage a specific image or images in a specific organization.

NOTE

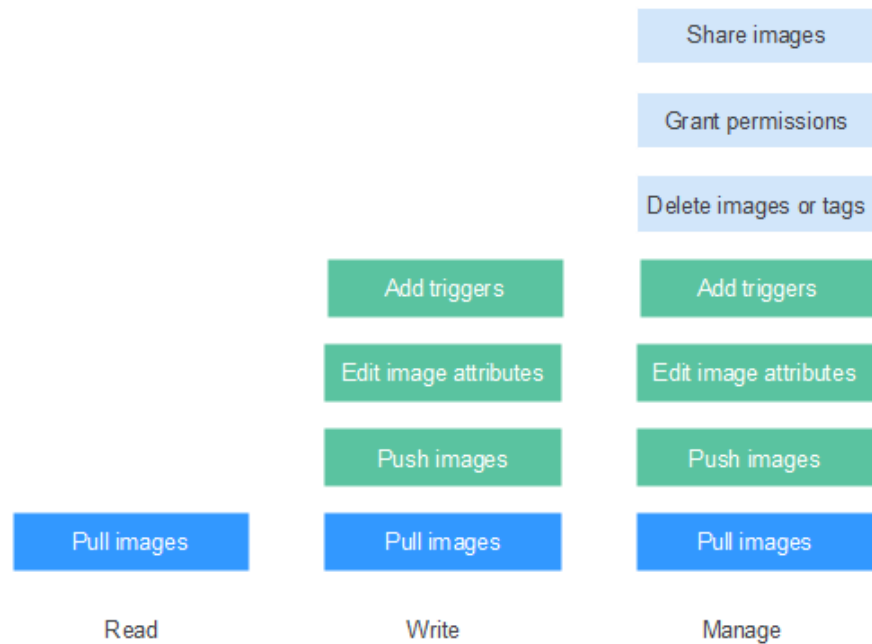
- An SWR account admin user is granted image management permission of all organizations by default, even if the user is not in the authorized user list of the organizations.

Authorization Method

You can grant permissions to users in SWR by using either of the following methods:

- [Grant permissions on an image details page](#) to allow users to read, write, and manage a specific image.
- [Grant permissions on an organization details page](#) to allow users to read, write, and manage all the images in an organization.

Figure 7-1 User permissions



You can grant the following three types of permissions to users:

- Read: Users can only pull images.
- Write: Users can pull and push images, add triggers, and edit image attributes.
- Manage: Users can pull and push images, delete images or tags, edit image attributes, grant permissions, add triggers, and share images with other users.

NOTE

To upload images to an organization, you require the write or manage permission for the organization to which images are uploaded. Write and manage permissions added on the image details pages will not be sufficient to upload images.

Granting Permissions for a Specific Image

To allow users to read, write, and manage a specific image, grant corresponding permissions to them on the details page of this image.

- Step 1** Log in to the SWR console.
- Step 2** In the navigation pane, choose **My Images** and click the desired image.
- Step 3** On the image details page, click the **Permissions** tab.
- Step 4** Click **Add Permission**. On the page displayed, click **Read**, **Write**, or **Manage** in the row of the desired username. Click **OK** to confirm.

----End

Modifying or Deleting Permissions for a Specific Image

You can also modify or delete user permissions on the image details page.

- To modify permissions, click **Modify** in the row of the desired username on the **Permissions** tab page. Select a permission in the **Permission** drop-down list, and click **Save** in the **Operation** column.
- To delete permissions, click **Delete** in the row of the desired username on the **Permissions** tab page, and then click **Yes**.

Granting Permissions for an Organization

To allow users to read, write, and manage all the images in an organization, grant corresponding permissions to them on the details page of this organization.

Only users with the **Manage** permission can grant permissions for other users.

Step 1 Log in to the SWR console.

Step 2 In the navigation pane, choose **Organization Management**. Then click **Details** in the row of the desired organization.

Step 3 On the **Users** tab page, click **Add Permission**. In the dialog box displayed, select permissions for users and click **OK**.

----End

Modifying or Deleting Permissions for an Organization

You can also modify and delete user permissions of an organization.

- To modify permissions, click **Modify** in the row of the desired username on the **Users** tab page. Select a permission in the **Permission** drop-down list, and click **Save** in the **Operation** column.
- To delete permissions, click **Delete** in the row of the desired username on the **users** tab page, and then click **Yes**.

8 FAQs

8.1 General FAQs

8.1.1 What Is SWR?

SoftWare Repository for Container (SWR) allows users to easily manage the full lifecycle of container images and facilitates secure deployment of images for your applications.

8.1.2 About SWR

How Many Images Can Be Stored in SWR?

SWR has no limit on the number of images. You can upload any number of images.

What Is the Bandwidth of SWR?

The bandwidth of SWR dynamically changes based on actual usage.

Does SWR Support Querying the CPU Architecture (x86 or Arm) of an Image?

- For a public image, you can log in to the SWR console, go to the image center, search for the target image, and view its details, including the architectures supported by the image.
- For a private image, you can Run **docker inspect** [*Image name:Version name*] to query the image architecture.

Example: docker inspect openjdk:7.

Figure 8-1 Example

```

}
  "Architecture": "amd64",
  "Os": "Linux",
  "Size": 621209007,
  "VirtualSize": 621209007,
  "GraphDriver": {
    "Data": {
      "LowerDir": "/var/lib/docker/overlay2/93ff8e16f44919ca8ec9e5c5077ea166c79a3f5f1dbf46288390a77d32cedf7b/diff:/var
/var/lib/docker/overlay2/579337e171a09f26649010fadac542b9b050079fda9a97837450c20ebc36076e/diff:/var/lib/docker/overlay2/d9b3e45a1339
74fc000a8a78c4462f066b56ba3d3af23fa4ba59cc1cf6efafb2/diff",
      "MergedDir": "/var/lib/docker/overlay2/09ead26ee61e1ecfd8556963d768e888a63b8f51dc506b4c806c7480e1b76852/merged",
      "UpperDir": "/var/lib/docker/overlay2/09ead26ee61e1ecfd8556963d768e888a63b8f51dc506b4c806c7480e1b76852/diff",
      "WorkDir": "/var/lib/docker/overlay2/09ead26ee61e1ecfd8556963d768e888a63b8f51dc506b4c806c7480e1b76852/work"
    }
  },
  "Name": "overlay2"
},
"RootFS": {
  "Type": "layers",
  "Layers": [
    "sha256:174f5685490326fc8a1c0f5570b8663732189b327007e47ff13d2ca59673db02",
    "sha256:fd431c3bcfb54e46ee912e04becffa279cd866c278cfff74dd0a15fca05dcdb9b",
    "sha256:f8200dd2ca0ff94a8beacb8df885ff2c9c4843dac363abea052181063b535445",
    "sha256:0f137c758756a06bc5b64d05ceb636525f1267fbef49b8147f651a062b54ea7"
  ]
}
}

```

8.1.3 How Do I Create a Container Image?

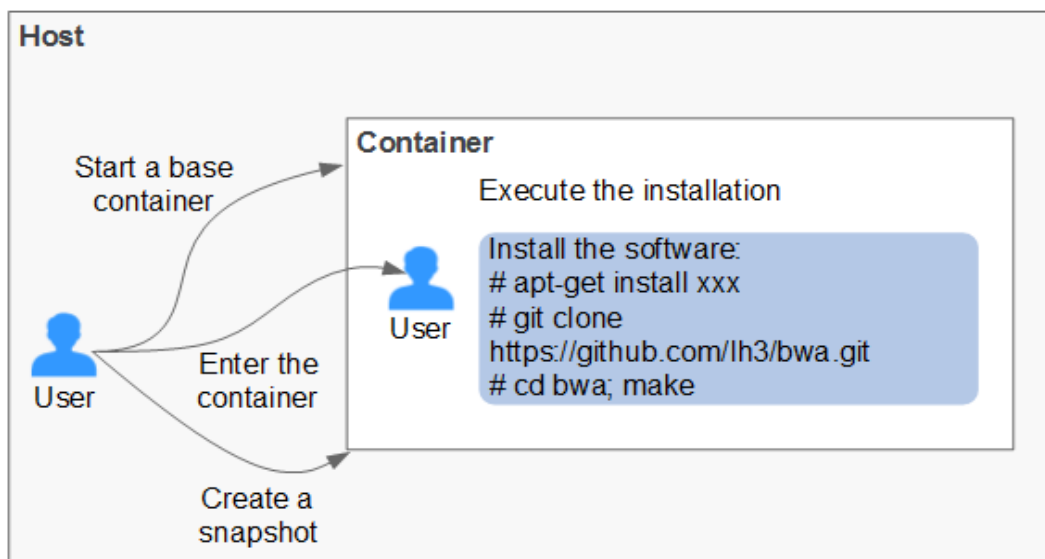
The following two approaches are for you to consider. Approach 1 is for images that will only be updated occasionally whereas approach 2 is for images that will be frequently updated.

- Approach 1: creating a snapshot. This approach involves three key steps: (1) Start a base container by running a base image (for example, Ubuntu image); (2) install the container engine software inside the base container; (3) create a snapshot of the container.
- Approach 2: creating a Dockerfile to build an image. This approach involves two key steps: (1) Write software installation instructions into a Dockerfile; (2) run the **docker build** command to build an image from the Dockerfile.

Approach 1: Creating a Snapshot

This approach is suitable for images that will only be updated occasionally.

Figure 8-2 Creating a snapshot



Procedure:

1. Install the container engine software on a host.
2. Start an empty base container in the interactive mode.
For example, start a CentOS container in the interactive mode.

docker run -it centos

3. Run the following commands to install the target software:

yum install XXX

git clone https://github.com/lh3/bwa.git

cd bwa;make

NOTE

Install Git in advance and check whether an SSH key is set on the local host.

4. Run the **exit** command to exit the container.

5. Create a snapshot.

docker commit -m "xx" -a "test" container-id test/image:tag

- **-a:** indicates the author of the base image.
- **container-id:** indicates the ID of the container you have started in step 2. You can run the **docker ps -a** command to query the container ID.
- **-m:** indicates the commit message.
- **test/image:tag:** indicates the repository name/image name:tag name.

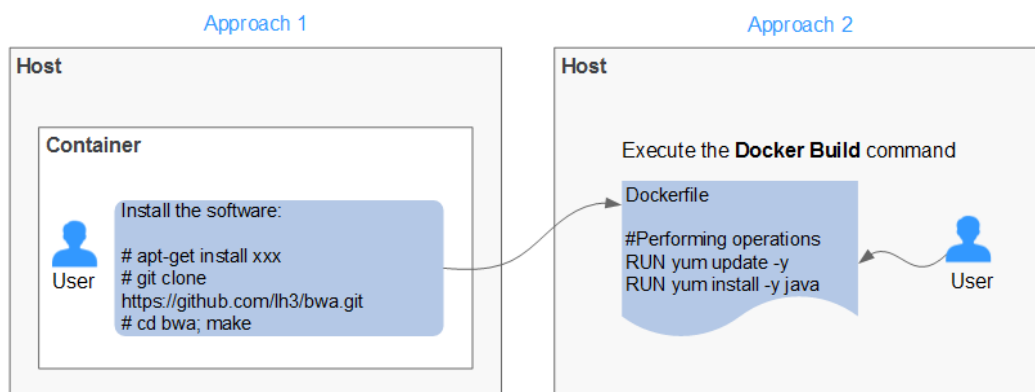
6. Run the **docker images** command to list the built container image.

Approach 2: Creating a Dockerfile to Build an Image

This approach is suitable for images that will be frequently updated. In [Approach 1](#), you create a snapshot of the whole container. This could be demanding if you need to frequently update your images. In this case, [Approach 2](#) is put forward to automate the image build process.

The idea behind [Approach 2](#) is to write the process of [Approach 1](#) into a Dockerfile and then run the **docker build -t test/image:tag** command to automatically build an image from the Dockerfile. In the preceding command, **.** indicates the path to the Dockerfile.

Figure 8-3 Creating a Dockerfile to build an image



Example Dockerfile:

NOTE

If an external network is required, ensure that network connectivity is available.

```
#Version 1.0.1
FROM centos:latest

# Setting the root user as the executor of subsequent commands
USER root

# Performing operations
RUN yum update -y
RUN yum install -y java

# Using && to concatenate commands
RUN touch test.txt && echo "abc" >>abc.txt

# Setting an externally exposed port
EXPOSE 80 8080 1038

# Adding a network file
ADD https://www.baidu.com/img/bd_logo1.png /opt/

# Setting an environment variable
ENV WEBAPP_PORT=9090

# Setting a work directory
WORKDIR /opt/

# Setting a start command
ENTRYPOINT ["ls"]

# Setting start parameters
CMD ["-a", "-l"]

# Setting a volume
VOLUME ["/data", "/var/www"]

# Setting the trigger operation for a sub-image
ONBUILD ADD ./app/src
ONBUILD RUN echo "on build excuted" >> onbuild.txt
```

Basic Syntax of Dockerfile

- FROM:
It is used to specify the parent image (base image) from which you are building a new image. Except annotations, a Dockerfile must start with a FROM instruction. Subsequent instructions run in this parent image environment until the next FROM instruction appears. You can create multiple images in the same Dockerfile by adding multiple FROM instructions.
- MAINTAINER:
It is used to specify the information about the author who creates an image, including the username and email address. This parameter is optional.
- RUN:
It is used to modify an image. Generally, RUN commands are executed to install libraries, and install and configure programs. After a RUN command is executed, an image layer will be created on the current image. The next command will be executed on the new image. The RUN statement can be in one of the following formats:
 - **RUN yum update:** Command that is executed in the **/bin/sh** directory.

- **RUN ["yum", "update"]**: Directly invoke **exec**.
- **RUN yum update && yum install nginx**: Use **&&** to connect multiple commands to a RUN statement.
- **EXPOSE**:
It is used to specify one or more network ports that will be exposed on a container. If there are multiple ports, separate them by spaces.
When running a container, you can set **-P** (uppercase) to map the ports specified in EXPOSE to random ports on a host. Other containers or hosts can communicate with the container through the ports on the host.
You can also use **-p** (lowercase) to expose the ports that are not listed in EXPOSE.
- **ADD**:
It is used to add a file to a new image. The file can be a host file, a network file, or a folder.
 - First parameter: source file (folder)
 - If a relative path is used, this path must correspond to the directory where the Dockerfile is located.
 - If a URL is used, the file needs to be downloaded first and then added to the image.
 - Second parameter: target path
 - If the source file is in the .zip or .tar file, the container engine decompresses the file and then adds it to the specified location of the image.
 - If the source file is a compressed network file specified by a URL, the file will not be decompressed.
- **VOLUME**:
It is used to create a mount point for a specified path (file or folder) in the image. Multiple containers can share data through the same mount point. Even if one of the containers is stopped, the mount point can still be accessed.
- **WORKDIR**:
It is used to specify a new work directory for the next command. The directory can be an absolute or a relative directory. WORKDIR can be specified multiple times as required. When a container is started, the directory specified by the last WORKDIR command is used as the current work directory of the container.
- **ENV**:
It is used to set an environment variable for running the container. When running the container, you can set **-e** to modify the environment variable or add other environment variables.
Example:
docker run -e WEBAPP_PORT=8000 -e WEBAPP_HOST=www.example.com ...
- **CMD**:
It is used to specify the default command for starting a container.

- **ENTRYPOINT:**

It is used to specify the default command for starting a container. Difference: For ENTRYPOINT, parameters added to the image during container running will be spliced. For CMD, these parameters will be overwritten.

 - If the Dockerfile specifies that the default command for starting a container is **ls -l**, the default command **ls -l** will be run accordingly. For example:
 - **ENTRYPOINT ["ls", "-l"]:** The program and parameter for starting a container are set to be **ls** and **-l** respectively.
 - **docker run centos:** The **docker run centos ls -l** command is run by default for starting a CentOS container.
 - **docker run centos -a:** When the **-a** parameter is added for starting a CentOS container, the **docker run centos ls -l -a** command is run by default.
 - If the Dockerfile specifies that the default command for starting a container is **--entrypoint** but you need to replace the default command, you can add **--entrypoint** parameters to replace the configuration specified in Dockerfile. Example:
docker run gutianlangyu/test --entrypoint echo "hello world"
- **USER:**

It is used to specify the user or UID for running the container, and running the RUN, CMD, or ENTRYPOINT command.
- **ONBUILD:**

Trigger command. During image build, the image builder of the container engine saves all commands specified by the ONBUILD command to the image metadata. These commands will not be executed in the process of building the current image. These commands will be executed only when a new image uses the FROM instruction to specify the parent image as the current image. Using the FROM instruction to build a child image based on the parent image created by the Dockerfile:
ONBUILD ADD. /app/src: The **ADD. /app/src** command is automatically executed.

8.1.4 How Do I Create an Image Package?

Run the **docker save** command to compress the container image into a .tar or .tar.gz package. The command format is as follows:

docker save [OPTIONS] IMAGE [IMAGE...]

[OPTIONS] can be set to **--output** or **-o**, indicating that the image is exported to a file.

Example:

```
$ docker save nginx:latest > nginx.tar
$ ls -sh nginx.tar
108M nginx.tar

$ docker save php:5-apache > php.tar.gz
$ ls -sh php.tar.gz
```

```
372M php.tar.gz
$ docker save --output nginx.tar nginx
$ ls -sh nginx.tar
108M nginx.tar

$ docker save -o nginx-all.tar nginx
$ docker save -o nginx-latest.tar nginx:latest
```

8.1.5 Are There Quotas for SWR Resources?

No quotas are imposed on SWR images. You can push as many images as you need.

Quotas are imposed on the number of organizations a user can create, as shown in [Table 8-1](#).

Table 8-1 SWR resource quotas

Resource Type	Quota
Organization	5

8.1.6 Why Does Organization Creation Fail?

Symptom: The creation of an organization fails, and a message is displayed indicating that the organization already exists. However, the organization is not found on the **Organizations** page.

Solution: Change the organization name to one which is globally unique in the Region.

If a message is displayed indicating that the organization already exists, the organization name may have been used by another user. Use another organization name.

8.2 Login Issues

8.2.1 What Are the Differences Between Long-Term Valid Login Commands and Temporary Login Commands?

- Temporary login commands will expire after 6 hours. A temporary login command can be used in scenarios such as temporary use and one-time external authorization. For production clusters that have high security requirements, it can also be used with periodic refresh.
- Long-term valid login commands are permanently valid. A long-term valid login command can be used in scenarios such as preliminary tests, CI/CD pipelines, and image pull from container clusters.

After you obtain a long-term valid login command, your temporary login commands will still be valid as long as they are in their validity periods.

The long-term valid and temporary login commands can be used by multiple users to log in to the system at the same time.

8.2.2 What Do I Do If My Login Command Expired?

Clear your browser cache and generate a new login command. The new login command will expire in six hours.

8.3 Pushing an Image

How Do I Push an Image to SWR by Calling APIs?

Currently, SWR does not provide APIs for image push. You can push images using the **docker push** command on a client or using the SWR console.

Why Is an Image Pushed Using a Container Engine Client to SWR Different in Size From One Uploaded Through the SWR Console?

Symptom

Assume that a nginx image of v2.0.0 is created on the local Docker client. The **docker images** command is run to query **SIZE** of the image. The size is **22.8 MB**.

```
$ docker images
REPOSITORY          TAG          IMAGE ID       CREATED        SIZE
nginx               v2.0.0      22f2bf2e2b4f  9 days ago    22.8MB
```

1. Run the **docker push** command to upload the image to SWR. The size of the image is **9.5 MB**.
2. On the local Docker client, pack the image into a **.tar** package. Download the **nginx.tar** package to the local host, and upload the package to SWR. The size of the package is **23.2 MB**.

The size of the image pushed through the client is different from that of the image uploaded through the SWR console.

Cause Analysis

Image layers are compressed into TGZ packages when images are pushed to SWR using a container engine client, whereas when they are uploaded through the SWR console, they are only packed without being compressed. Therefore, the same image will be of different sizes when it is uploaded in these two different ways.

Can I Push Arm-based Container Images to SWR?

SWR has no restriction on the kernel architecture of images. There is no difference between pushing an Arm-based image and an x86-based image to SWR.

What Protocol Is Used to Push Images to SWR When I Run the docker push Command?

HTTPS is used.

Will an Image Be Overwritten If I Push an Image That Have the Same Name and Tag with it?

Yes, the original image will be overwritten.

What Is the Maximum Size of an SWR Layer?

If you use the container engine client to push images to SWR, each image layer cannot exceed 10 GB.

What Is the Rate Limit for a Tenant to Push Images over the Internet?

To avoid mutual interference between tenants when they push SWR images, the image push traffic for a single tenant is limited to 20 QPS. The traffic exceeding this value will be blocked. In this case, Docker will receive 503 and automatically retry traffic control requests.

Does SWR Support Resumable Image Push?

No.

8.4 Pulling an Image

How Do I Pull an Image from SWR by Calling APIs?

Currently, SWR does not provide APIs for image pull. You can pull images using the **docker push** command on a client.

Where Are the Images Pulled by docker pull Stored?

Images pulled by running the **docker pull** command are stored on your local hosts. You can run the **docker save** command to save images into TAR archive files.

Can I Pull Images from Another Region?

Cross-region image pull over public network is supported.

Cross-region image pull over public network is supported. Ensure that you have obtained the correct login command.

Can I Pull Images on the SWR Console to a Local PC?

Images stored in SWR cannot be directly downloaded through the console. You can perform the following operations to pull the images:

1. Obtain the image pull command on the image details page.
2. Run the obtained command on the device where the Docker client is installed.
3. Save the image as a TAR or TAR.GZ file.

Example:

```
docker save nginx:v1 > nginx.tar
```

4. Download the file to the local host.

What Are the Possible Causes of Slow Image Pull?

1. The network is abnormal.
2. There are multiple image layers.
3. Image pull tasks are serially executed. A task cannot be executed until the previous task is complete. The timeout interval for a pull task is 30 minutes.

Will a Pulled Image Overwrite an Existing Image with the Same Name and Tag?

If they have the same manifest, the existing image will not be overwritten. Otherwise, it will be overwritten.

8.5 Image Management FAQs

8.5.1 Can I Pull Images from SWR to a Local PC?

Container images cannot be downloaded from the SWR console, but you can run commands to pull them.

1. Obtain the image pull command on the image details page.
2. Run the pull command on the Docker client.

Example:

```
docker pull {Image-repository-address}/group/nginx:v1
```

3. Save the image as a tar or tar.gz package.

Example:

```
docker save nginx:v1 > nginx.tar
```

4. Download the package to your local PC.

8.6 Troubleshooting

8.6.1 Why Does the Login Command Fail to Be Executed?

Possible causes are as follows:

1. The container engine is not properly installed, in which case the following error is reported:

docker: command not found


Solution: Reinstall the container engine.

- To use SWR, the Docker version must be between 1.11.2 (included) and 24.0.9 (included).
- If the container engine client is in a private network, bind an elastic IP address (EIP) to the client. This EIP will allow the client to download installation packages from the website.

2. The temporary login command has expired, or the regional project name, AK, or login key in the command is invalid. Error:

unauthorized: authentication required

Solution: Log in to the SWR console. In the navigation pane on the left, choose **My Images**. On the page displayed, click **Upload Through Client**. Then you can find the information on how to obtain a login command.

- a. To obtain a temporary login command, click **Generate a temporary login command** and then click  to copy the command.
 - b. To obtain a long-term valid login command, click **learn how to obtain a login command that has long-term validity** and follow the instructions.
3. The image repository address in the login command is incorrect, in which case the following error is reported:

Error logging in to v2 endpoint, trying next endpoint: Get https://{{endpoint}}/v2/: dial tcp: lookup {{endpoint}} on xxx.xxx.xxx.xxx:53 : no such host

Solutions:

- a. Change the image repository address in the login command.
- b. Generate a temporary login command. For detailed instructions, see [2](#).

4. **x509: certificate has expired or is not yet valid**

The preceding error is reported when the AK/SK in the login command with long-term validity is deleted. In this case, use a valid AK/SK to generate a login command.

5. **x509: certificate signed by unknown authority**

Possible Causes:

The container engine client communicates with SWR through HTTPS. The client verifies the server certificate. If the server certificate is not issued by an authoritative organization, the following error message is displayed: "x509: certificate signed by unknown authority"

Solutions:

If you trust the server and skip certificate authentication, manually configure Docker startup parameters as follows:

- CentOS:

Modify the **/etc/docker/daemon.json** file. If the file does not exist, manually create it. Add the following content to the file:

```
{  
  "insecure-registries": ["{Image repository address}"]  
}
```

- Ubuntu:

Modify the **/etc/default/docker** file and add the following content to **DOCKER_OPTS**:

```
DOCKER_OPTS="--insecure-registry {image repository address}"
```

- EulerOS:

Modify the **/etc/sysconfig/docker** file and add the following content to **INSECURE_REGISTRY**:

```
INSECURE_REGISTRY='--insecure-registry {image repository address}'
```

 NOTE

The image repository address can be a domain name or an IP address.

- To obtain the image repository address in domain name format, obtain a temporary login command by referring to 2. The domain name at the end of the command is the image repository address.
- To obtain the image repository address in IP address format, ping the image repository address in the domain name format.

After the configuration, run the **systemctl restart docker** command to restart the container engine.

6. **denied: Not allow to login, upload or download image**

If you concurrently upload large numbers of images or frequently request access to the service, the system will blacklist you. As a result, you cannot log in to the system or upload or download images. Try again 30 minutes later.

8.6.2 Why Does an Image Fail to Be Pushed Through a Container Engine Client?

denied: you do not have the permission

Problem: When you push an image to SWR through your container engine client, the operation fails with the following information returned.

denied: you do not have the permission

Possible Causes:

- The organization name you specified has already been used by another user or the maximum number of organizations that you are allowed to create has been reached.
- The **docker login** command you used to log in to SWR is generated using the AK and SK of an IAM user who does not have the permission of the target organization.

Solutions:

- If the organization name has been registered by another user, create an organization and then upload the image.
- If the number of SWR organizations has reached the quota (5 per user), push the image to an existing organization.
- If the IAM user does not have the permission of the target organization, log in using the cloud account and grant the permission to the IAM user. Then, try again.

Message "tag does not exist: xxxxxx" or "An image does not exist locally with the tag: xxxxxx" Displayed

Problem: When you push an image to SWR through your container engine client, the operation fails with the following information returned.

tag does not exist: xxxxxx

Or

An image does not exist locally with the tag: xxxxxx

Possible cause: The image or image tag to be pushed does not exist.

Solution: Run the **docker images** command to view all the local images. Check the target image name and tag, and push the image again.

name invalid: 'repository' is invalid

Problem: When you push an image to SWR through your container engine client, the operation fails with the following information returned.

name invalid: 'repository' is invalid

Possible cause: The organization name or image name does not comply with the naming rules.

Solution: The regular expressions of the organization (namespace) name and image (repository) name are as follows:

namespace: The value contains a maximum of 64 characters and must meet regular expression `^[a-z]+(?::(?:[_]|[-]*)[a-z0-9]+)+)?$`.

repository: The value contains a maximum of 128 characters and must meet regular expression `^[a-z0-9]+(?::(?:[_]|[-]*)[a-z0-9]+)+)?$`.

Specify a valid organization name or image name, and push the image again.

denied: Not allow to login, upload or download image

Symptom: When you push an image to SWR through your container engine client, the operation fails with the following error message:

Not allow to login, upload or download image

Possible cause: You concurrently uploaded large numbers of images or frequently requested access to the service, and the system blacklisted you.

Solution: Try again 30 minutes later.

Image Push Occasionally Times Out

Problem: Image push occasionally times out.

Solution: When you push an image from a server in Chinese mainland to a server outside Chinese mainland, the network may be unstable.

8.6.3 Why Does an Image Fail to Be Uploaded Through SWR Console?

SWR has strict requirements on image name and address format. Invalid image names or addresses could lead to upload failures.

Invalid Image Format

Problem: When you upload an image to SWR through the SWR console, an error message is displayed, indicating that the image format is invalid.

Possible cause: Invalid image address leads to upload failure.

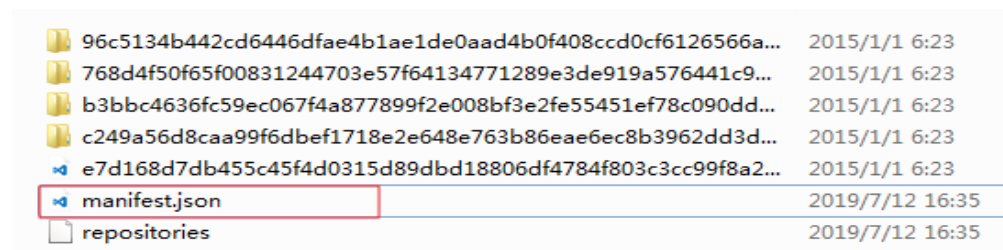
The image tag, which is at the end of an image address, can be omitted. When it is omitted, the latest version of the image will be pushed. Other parts of the image address cannot be omitted. Ensure that they are all correctly configured.

Example: registry.example.com/repo_namespace/repo_name:tag

- **registry.example.com** is an example address of the SWR image repository. Replace it with the actual address.
- **repo_namespace:** organization name. It contains a maximum of 64 characters and must meet regular expression $^([a-z]+(?:[:_]|[-]*)[a-z0-9]+)+?)$$.
- **repo_name:tag:** image name and tag. The image name contains a maximum of 128 characters and must meet regular expression $^([a-z0-9]+(?:[:_]|[-]*)[a-z0-9]+)+?)$$.

To view the image address, decompress the image. Open the **manifest.json** file, and check the value of **RepoTags**.

Figure 8-4 Files in the decompressed package



96c5134b442cd6446dfae4b1ae1de0aad4b0f408ccd0cf6126566a...	2015/1/1 6:23
768d4f50f65f00831244703e57f64134771289e3de919a576441c9...	2015/1/1 6:23
b3bbc4636fc59ec067f4a877899f2e008bf3e2fe55451ef78c090dd...	2015/1/1 6:23
c249a56d8caa99f6dbef1718e2e648e763b86eae6ec8b3962dd3d...	2015/1/1 6:23
e7d168d7db455c45f4d0315d89dbd18806df4784f803c3cc99f8a2...	2015/1/1 6:23
manifest.json	2019/7/12 16:35
repositories	2019/7/12 16:35

Solution: Tag the image again according to the naming rules. Run the **docker save** command, and upload the image on the SWR console.

denied: Not allow to login, upload or download image

Symptom: When you push an image to SWR through the console, the operation fails with the following error message:

Not allow to login, upload or download image

Possible cause: You concurrently uploaded large numbers of images or frequently requested access to the service, and the system blacklisted you.

Solution: Try again 30 minutes later.

Stuck at the Upload Page Until It Times Out

Problem: When you upload an image to SWR through the SWR console, the upload progress is stuck and the upload task times out at the end.

Possible cause: Invalid image name leads to upload failures.

Solution: Modify the image name according to the naming rules, and try uploading the image again.

NOTICE

It is the image name in the **repositories** and **manifest.json** files that should be checked and modified rather than the name of the image file you select and upload on the SWR console.

8.6.4 Why Does the docker pull Command Fail to Be Executed?

x509: certificate signed by unknown authority

Problem: When you run the **docker pull** command to pull an image from SWR, error message "x509: certificate signed by unknown authority" is displayed.

Possible Causes:

- A container engine client and SWR communicate with each other using HTTPS. When the client verifies the server certificate and finds that the root certificate installed on the client is incomplete, the error message "x509: certificate signed by unknown authority" is displayed.
- A proxy is configured on the container engine client.

Solution:

- If you trust the server and skip certificate authentication, manually configure the startup parameters for the container engine using either of the following methods (use the actual image repository address):

- Add the following configuration to the **/etc/docker/daemon.json** file. If the file does not exist, manually create it. Ensure that two-space indents are used in the configuration.

```
{  
  "insecure-registries":["Image repository address"]  
}
```

- **/etc/sysconfig/docker:**

```
INSECURE_REGISTRY='--insecure-registry=Image repository address'
```

After configuration, run the **systemctl restart docker** or **service docker start** command to restart the container engine.

- Run the **docker info** command to check whether the proxy is correctly configured. If not, modify the configuration.

Error: remote trust data does not exist

Problem: When you run the **docker pull** command to pull an image from SWR, message "Error: remote trust data does not exist" is displayed.

Possible cause: The image signature verification is enabled on the client. However, the image to be pulled does not contain a signature layer.

Solution: Check whether the environment variable **DOCKER_CONTENT_TRUST** is set to **1**. If yes, delete **DOCKER_CONTENT_TRUST=1** from the **/etc/profile** file and run the **source /etc/profile** command to make the modification take effect.

denied: Not allow to login, upload or download image

Symptom: When you pull an image from SWR through your container engine client, the operation fails with the following error message:

Not allow to login, upload or download image

Possible cause: You concurrently uploaded large numbers of images or frequently requested access to the service, and the system blacklisted you.

Solution: Try again 30 minutes later.

8.6.5 What Do I Do If an Error Occurs When I Call an API?

If the URL of an SWR API contains an image name with a slash (/), an error may occur. Replace the slash with a dollar sign (\$).

For example, an image named **a/b** belongs to organization **c**.

When you call an API to view the image information, replace **GET /v2/manage/namespaces/c/repos/a/b** with **GET /v2/manage/namespaces/c/repos/a\$b**.

8.7 Other FAQs

8.7.1 Why Does a CCE Workload Cannot Pull an Image from SWR and the Message Indicating "Not Logged In" Is Displayed?

If a CCE workload cannot pull an SWR image and the message indicating "Not logged in" is displayed, check whether the YAML file of the workload contains the **imagePullSecrets** field and whether the value of **name** is fixed to **default-secret**.

Example:

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - image: nginx
          imagePullPolicy: Always
          name: nginx
          imagePullSecrets:
            - name: default-secret
```

8.7.2 Why Is an Image Pushed Using a Container Engine Client to SWR Different in Size From One Uploaded Through the SWR Console?

Symptom

Assume that a nginx image of v5 is created on the local Docker client. The **docker images** command is run to query **SIZE** of the image. The size is 22.8 MB.

```
$ docker images
REPOSITORY          TAG          IMAGE ID      CREATED      SIZE
nginx               v5          22f2bf2e2b4f 9 days ago  22.8MB
```

1. Run the **docker push** command to push the image to SWR. The size of the image is 9.5 MB.

Tag	Size	Image Pull Command	Updated
v5	9.5 MB	docker pull swr.cn-north-7.myhuaweicloud.com/zgy/test-image:v5	2024/05/10 18:10:16 GMT+08:00

2. On the local Docker client, pack the image into a TAR package. Download the **nginx.tar** package to the local host, and upload the package to SWR. The size of the package is 23.2 MB.

Tag	Size	Image Pull Command	Updated
v5	23.2 MB	docker pull swr.cn-north-7	2024/05/11 09:29:01 GMT+08:00

The size of the image pushed through the client is different from that of the image uploaded through the SWR console.

Possible Cause

Image layers are compressed into TGZ packages when images are pushed to SWR using a container engine client, whereas when they are uploaded through the SWR console, they are only packed without being compressed. Therefore, the same image will be of different sizes when it is uploaded in these two different ways.

8.7.3 How Many Tenants Can I Share an SWR Private Image With?

500