

SoftWare Repository for Container

FAQs

Issue 01
Date 2022-09-30



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

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

Trademarks and Permissions



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.

Contents

1 General FAQs.....	1
1.1 What Is SWR?.....	1
1.2 About SWR.....	1
1.3 How Do I Create a Container Image?.....	2
1.4 How Do I Create an Image Package?.....	6
1.5 Are There Quotas for SWR Resources?.....	7
1.6 Why Does Organization Creation Fail?.....	7
2 Image Management FAQs.....	8
2.1 Image Push and Pull.....	8
2.2 How Many Tenants Can I Share an SWR Private Image with?.....	9
2.3 What Are the Differences Between Long-Term Valid Login Commands and Temporary Login Commands?.....	9
2.4 Why Is an Image Uploaded Through the Client to SWR Different in Size From One Uploaded Through the SWR Console?.....	9
2.5 Can I Pull Container Images on the SWR Console to a Local PC?.....	10
3 Troubleshooting.....	11
3.1 Why Does the Login Command Fail to Be Executed?.....	11
3.2 Why Does an Image Fail to Be Uploaded Through a Container Engine Client?.....	13
3.3 Why Does an Image Fail to Be Uploaded Through SWR Console?.....	14
3.4 Why Does the docker pull Command Fail to Be Executed?.....	15
4 Other FAQs.....	17
4.1 Why Does a CCE Workload Cannot Pull an Image from SWR and the Message Indicating "Not Logged In" Is Displayed?.....	17
4.2 How Do I Obtain the Docker Image of the Target Software?.....	18

1 General FAQs

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.

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.

Is SWR Charged?

The billing items of SWR include storage space and traffic. Currently, it is free of charge.

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 1-1 Example

```
},
  "Architecture": "amd64",
  "OS": "Linux",
  "Size": 621209007,
  "VirtualSize": 621209007,
  "GraphDriver": {
    "Data": {
      "LowerDir": "/var/lib/docker/overlay2/93ff8e16f44919ca8ec9e5c5077ea166c79a3f5f1dbf46288390a77d32cedf7b/diff:/var/lib/docker/overlay2/579337e171a09f26649010fadac542b9b058079fda9a97837450c20ebc36076e/diff:/var/lib/docker/overlay2/d9b3e45a133974fc0b0a8a78c4462f066b56ba3d3af23fa4ba59cc1cf6efafb2/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:fd31e3bcfb54e46ee912e04becffa279cd866c278cfff74dd0a15ca05dcdb9b",
    "sha256:f020dd2ca0ff94a8beacb8df885ff2c9c4843dac363abea052181063b535445",
    "sha256:0f137c758756a06bc5bd64d05ceb636525f1267fbef49b8147f651a062b54ea7"
  ]
}
```

1.3 How Do I Create a Container Image?

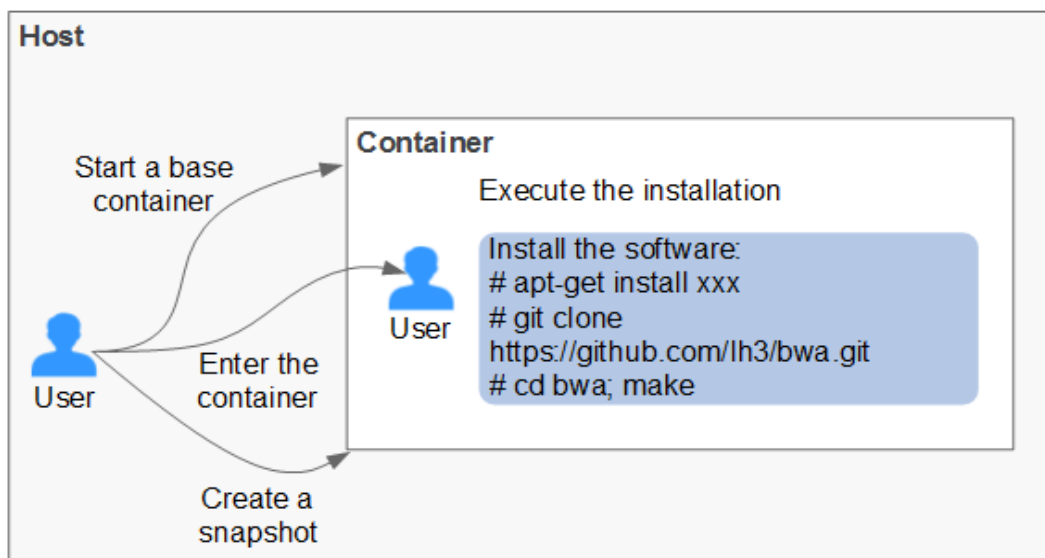
The following two approaches are for you to consider. Approach 1 is for images that will only be updated occasionally whereas approach 2 is for images that will be frequently updated.

- 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 1-2 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.

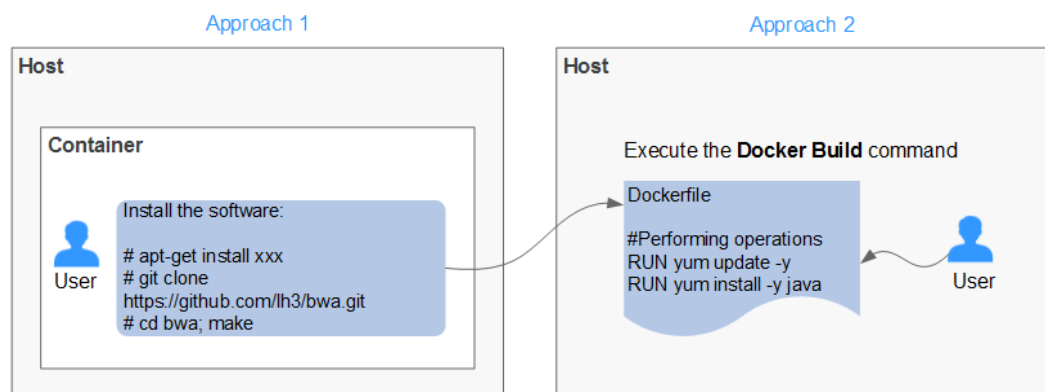
6. Run the **docker images** command to list the built container image.

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 1-3 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.

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
```

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 1-1](#).

Table 1-1 SWR resource quotas

Resource Type	Quota
Organization	5

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.

2 Image Management FAQs

2.1 Image Push and Pull

How Do I Push an Image to SWR Through 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.

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 pull** command on a client.

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.

Where Are the Images Pulled by Running the **docker pull** Command 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.

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.

Can SWR Be Accessed over Private Networks? Will I Be Charged for Pushing and Pulling Images over Private Networks?

If your machine and the image repository are in the same region, you can access the image repository through private networks. No additional fees are charged for private network access because you have paid for your servers and EIPs.

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.

2.2 How Many Tenants Can I Share an SWR Private Image with?

500

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

- Temporary login commands will expire after 24 hours.
- Long-term valid login commands are permanently valid.

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.

2.4 Why Is an Image Uploaded Through the 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 uploaded 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 uploaded to SWR through the 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.

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

Container 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.

Example:

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

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.

3 Troubleshooting

3.1 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.

- It is advised to install container engine 1.11.2 or later because earlier versions do not support image push to SWR.
- 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, access key (AK), or login key in the command is incorrect, in which case the following error is reported:

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"

Figure 3-1 Error x509

```
[root@luster01-fd0rb ~]# docker login -u cn-north-1@BAMUNTAIXIUBUGFF -p 5cad07ba37badb2956c29f5fee1173abe06f226fa509f7ab14dc85d749599d2 registry.cn-north-1.huaweiclouds.com
Error response from daemon: Get https://registry.cn-north-1.huaweiclouds.com/v2/users/: 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.

3.2 Why Does an Image Fail to Be Uploaded 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 reaches the quota (5 per user), upload the image to an existing organization.
- If the IAM user does not have the permission of the target organization, log in as the cloud account and grant corresponding permissions to the IAM user and 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.

3.3 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.

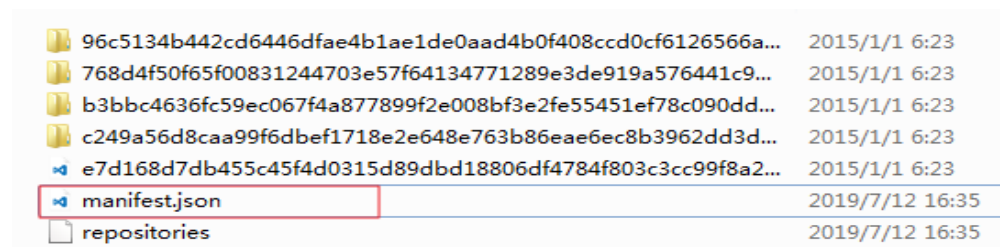
The image tag, which is at the end of an image address, can be omitted. When it is omitted, the latest version of the image will be pushed. Other parts of the image address cannot be omitted. Ensure that they are all correctly configured.

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 3-2 Files in the decompressed package



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

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.

3.4 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 certificates" 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 certificates" 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.

4 Other FAQs

4.1 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
```

4.2 How Do I Obtain the Docker Image of the Target Software?

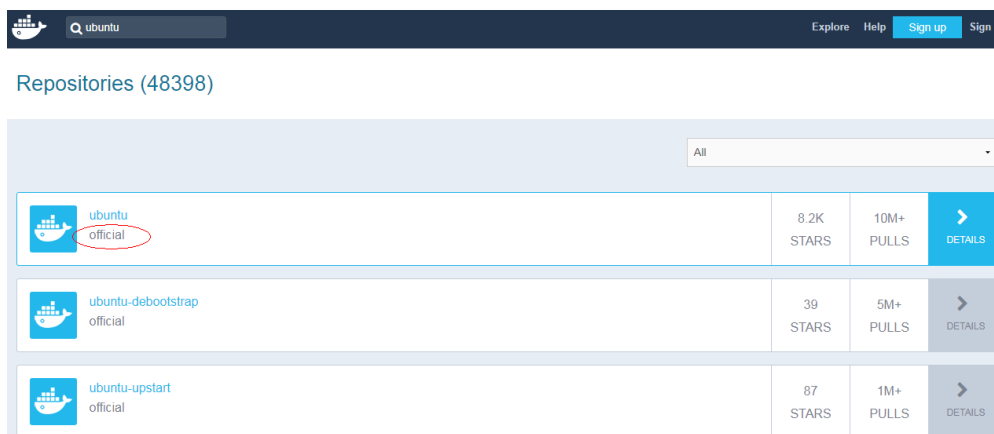
Searching Docker Hub for a Target Docker Image

Docker Hub provides more than 400,000 public Docker images for downloading various software, and the number keeps increasing at a rate of 5,000 per week. Therefore, you can find the corresponding image version except for the software developed by yourself. The Docker Hub address is <https://hub.docker.com/>.

You are advised to obtain certified images of the following software directly from Docker Hub, rather than building them from scratch.

- **Operating systems**

For example, Ubuntu, SUSE, and CentOS.

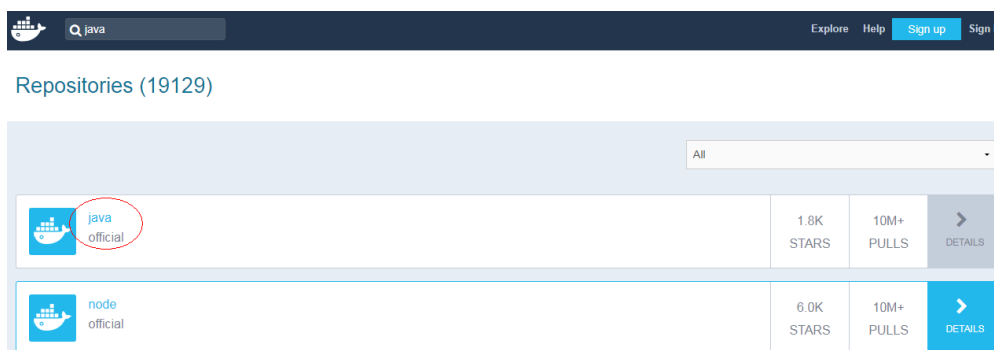


The screenshot shows the Docker Hub search results for 'ubuntu'. The search bar contains 'ubuntu' and the results are filtered to 'All'. The top result is 'ubuntu/official' with 8.2K stars and 10M+ pulls. Other results include 'ubuntu-debootstrap/official' and 'ubuntu-upstart/official'.

Repository	Stars	Pulls	Details
ubuntu/official	8.2K	10M+	DETAILS
ubuntu-debootstrap/official	39	5M+	DETAILS
ubuntu-upstart/official	87	1M+	DETAILS

- **Basic programming languages**

For example, Java, Python, R, and Golang.

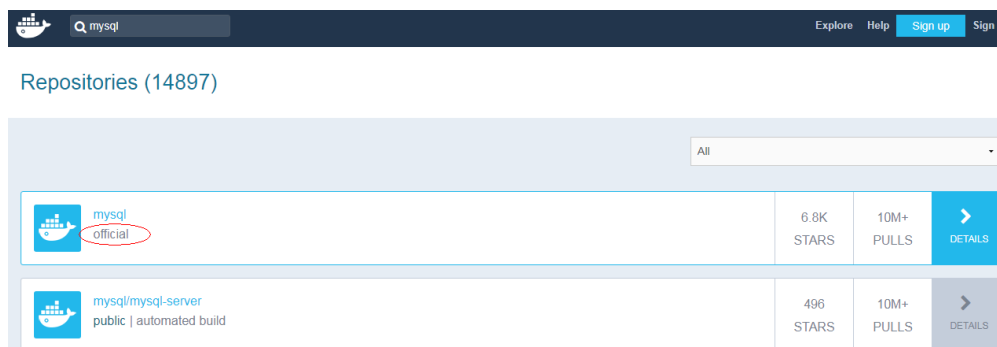


The screenshot shows the Docker Hub search results for 'java'. The search bar contains 'java' and the results are filtered to 'All'. The top result is 'java/official' with 1.8K stars and 10M+ pulls. Other results include 'node/official'.

Repository	Stars	Pulls	Details
java/official	1.8K	10M+	DETAILS
node/official	6.0K	10M+	DETAILS

- **Popular software**

For example, Tomcat, MySQL, and Nginx.



Searching for the Docker Image of the Target Software from Google

For software located in third-party image repositories, you can search for related images using Google. During the search, you only need to add Docker keywords next to software names.

Example:

