

# 3.Understand and publish images

---

## 3.Understand and publish images

- 3.1、 image understanding
- 3.2、 UnionFS (Federated file systems)
- 3.3、 image layering
  - 3.3.1、 hierarchical understanding
  - 3.3.2、 Docker images should use layering benefits
- 3.4、 Make and publish images
  - 3.4.1、 Make an image
  - 3.4.2、 Publish the image
    - 3.4.2.1、 Steps to publish the image to docker hub:
    - 3.4.2.2、 Steps to publish the image to tar package:

## 3.1、 image understanding

---

1. An image is a lightweight, executable stand-alone software package that contains everything needed to run a piece of software. We package applications and configurations into a ready, deliverable, deployable runtime environment, including code, libraries, environment variables and configuration files required for runtime, and this large package runtime environment is the image file.
2. Docker container instances can only be generated through image files.

## 3.2、 UnionFS (Federated file systems)

---

1.Union file system (UnionFS) is a hierarchical, lightweight, high-performance file system, it is the basis of docker images, and supports the modification of the file system as a commit to overlay layer by layer, while different directories can be mounted under the same virtual file system.

2.The image can be inherited through layering, and based on the basic image, various specific application images can be made.

Features of the Union file system: load multiple file systems at the same time, but from the outside, only one file system can be seen; Federated loading overlays the layers of file systems so that the final file system contains files and directories for all layers.

## 3.3、 image layering

---

When downloading an image, pay attention to the downloaded log output, you can see that it is downloading layer by layer:

```

jetson@ubuntu:~$ docker pull mysql
Using default tag: latest
latest: Pulling from library/mysql
6425367b44c9: Pull complete
7cef374d113a: Pull complete
1751ddbc0d77: Pull complete
f41e9e3c6d9a: Pull complete
c26e9c11cd2d: Pull complete
949ad8819238: Pull complete
3028a5ad3fd0: Pull complete
a41584bf2c82: Pull complete
f413abbd4b9d: Pull complete
da7c55c30cf5: Pull complete
038fc84e09b5: Pull complete
Digest: sha256:a43f6e7e7f3a5e5b90f857fbed4e3103ece771b19f0f75880f767cf66bbb6577
Status: Downloaded newer image for mysql:latest
docker.io/library/mysql:latest
jetson@ubuntu:~$

```

# The way to view image layering can be done with the command: `docker image inspect image name`

```
jetson@ubuntu:~$ docker image inspect mysql:latest
```

```

[
  {
    "Id":
"sha256:5371f8c3b63eec64a33b35530be5212d6148e0940111b57b689b5ba1ffe808c8",
    "RootFS": {
      "Type": "layers",
      "Layers": [
        "sha256:d6d4fc6aef875958d6186f85f03d88e6bb6484ab2dd56b30a79163baceff2f6d",
        "sha256:05c3b0b311a02bc56ca23105a76d16bc9b8c1d3e6eac808f4efb1a2e8350224b",
        "sha256:7b80f7f05642477ebc7d93de9539af27caab7c41a768db250fe3fe2b5506ca2c",
        "sha256:50e037faefab22cb1c75e60abb388b823e96a845650f3abd6d0a27e07a5a1d5e",
        "sha256:66040abb3f7201d2cc64531349a8225412db1029447a9431d59d999c941d56f6",
        "sha256:857162425652837a362aa5f1c3d4974cc83702728793de52ba48176d5367a89b",
        "sha256:7eebed3016f6b6ab68aa8e6be35f0689a3c18d331b7b542984a0050b859eaf26",
        "sha256:2fc4c142633d57d795edc0f3fd457f99a35fa611eab8b8c5d75c66e6eb729bc2",
        "sha256:7fde2d12d484f0c14dabd9ca845da0bcdaf60bd773a58ca2d73687473950e7fe",
        "sha256:9319848a00d38e15b754fa9dcd3b6e77ac8506850d32d8af493283131b9745a3",
        "sha256:5ff94d41f068ea5b52244393771471edb6a9a10f7a4ebafda9ef6629874a899b"
      ]
    },
    "Metadata": {
      "LastTagTime": "0001-01-01T00:00:00Z"
    }
  }
]

```

### 3.3.1、 hierarchical understanding

- All docker images start from a base image layer, and when modifications or additions are made, a new image layer will be created on top of the current image layer.
- For a simple example, if a new image is created based on Ubuntu 20.04, this is the first layer of the new image; If you add a Python package to the image, a second image layer is created on top of the base image layer; If you continue to add a security patch, a third mirror layer is created.
- Docker images are all read-only, and when the container starts, a new writable layer is loaded on top of the image! This layer is what we usually call the container layer, and what is under the container is called the image layer!

### 3.3.2、 Docker images should use layering benefits

Resource sharing, for example, if there are multiple images built from the same base image, then the host only needs to keep a base image on disk, and only one base image needs to be loaded in memory, so that all containers can be served, and each layer of the image can be shared.

## 3.4、 Make and publish images

### 3.4.1、 Make an image

Method 1: Submit an image from the container:

```
# command
docker commit -m="description of commit" -a="author" container id Target image
name to be created: [tag name] [You can also omit the -m -a parameter]

# test
jetson@ubuntu:~$ docker ps -a
CONTAINER ID   IMAGE          COMMAND        CREATED        STATUS        PORTS
NAMES
c54bf9efae47   ubuntu:latest  "/bin/bash"    3 hours ago    Up 24 minutes
funny_hugle

jetson@ubuntu:~$ docker commit c54bf9efae47 ubuntu:1.0
sha256:78ca7be949b6412f74ba12e8d16bd548aaa7c3fa25134326db3a67784f848f8f
jetson@ubuntu:~$ docker images      # Generated the ubuntu:1.0 image
REPOSITORY      TAG            IMAGE ID       CREATED        SIZE
ubuntu          1.0           78ca7be949b6   5 seconds ago  69.2MB
ubuntu          latest        bab8ce5c00ca   6 weeks ago    69.2MB
hello-world     latest        46331d942d63   13 months ago  9.14kB
```

```
jetson@ubuntu:~$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
ebdc5cc1469f   ubuntu:latest  "/bin/bash"             2 hours ago   Exited (0) 2 hours ago
a28343337ad3   ubuntu:latest  "/bin/bash"             2 hours ago   Exited (0) 2 hours ago
jetson@ubuntu:~$ docker commit ebdc5cc1469f ubuntu:1.0
sha256:43aeb49d903b2ab361486b223ad32d4b62ffc3bd4dffb72bcb30555bf7ace9a
jetson@ubuntu:~$ docker images
REPOSITORY    TAG          IMAGE ID          CREATED        SIZE
ubuntu        1.0          43aeb49d903b     7 seconds ago  69.2MB
192.168.2.51:5000/ro
s2-base       2.0.2       558e90afa763     3 days ago    5.61GB
192.168.2.51:5000/ro
s2-base       2.0.1       850d7fca6fbe     3 days ago    6.14GB
ubuntu        latest       6a47e077731f     2 weeks ago   69.2MB
jetson@ubuntu:~$
```

Method 2: Make an image from a dockerfile:

```
# command
docker build -f dockerfile file path -t new image name: TAG . # The docker build
command has a . Represents the current directory

# Test
docker build -f dockerfile-ros2 -t yahboomtechnology/ros-foxy:1.2 .

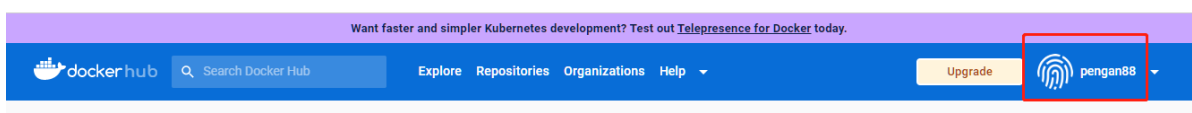
For more information on writing dockerfiles, please refer to :
https://docs.docker.com/develop/develop-images/dockerfile_best-practices/
```

## 3.4.2. Publish the image

The docker repository is a centralized storage place for image files. The largest public repository is Docker Hub (<https://hub.docker.com/>), which houses a huge number of images for users to download.

### 3.4.2.1. Steps to publish the image to docker hub:

- 1、Address: <https://hub.docker.com/> , register an account first
- 2、Ensure that the account can log in normally



- 3、Use the tag command to modify the image name

The specification for publishing an image to docker hub is:

```
docker push registration username/image name
```

For example, my registered username here is: pengan88, then you must first change the image name

```
# Command:
docker tag [image ID] [The name of the image after modified]

# test
jetson@ubuntu:~$ docker images
REPOSITORY    TAG          IMAGE ID          CREATED        SIZE
ubuntu        1.0          78ca7be949b6     5 seconds ago  69.2MB
ubuntu        latest       bab8ce5c00ca     6 weeks ago   69.2MB
```

```
hello-world          latest      46331d942d63    13 months ago    9.14kB
jetson@ubuntu:~$ docker tag 78ca7be949b6 pengan88/ubuntu:1.0
jetson@ubuntu:~$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
pengan88/ubuntu	1.0	78ca7be949b6	23 minutes ago	69.2MB
ubuntu	1.0	78ca7be949b6	23 minutes ago	69.2MB
ubuntu	latest	bab8ce5c00ca	6 weeks ago	69.2MB
hello-world	latest	46331d942d63	13 months ago	9.14kB



#### 4. Log in to Docker Hub to publish the image:

```
jetson@ubuntu:~$ docker login -u pengan88
Password: # Enter the password of the account registered by Docker Hub here
WARNING! Your password will be stored unencrypted in
/home/jetson/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store
Login Succeeded
```

```
jetson@ubuntu:~$ docker push pengan88/ubuntu:1.0
The push refers to repository [docker.io/pengan88/ubuntu]
ca774712d11b: Pushed
874b048c963a: Mounted from library/ubuntu
1.0: digest:
sha256:6767d7949e1c2c2adffbc5d3c232499435b95080a25884657fae366ccb71394d size:
736
```

#### 5. Visit Docker Hub to see that it has been successfully released


Want faster and simpler Kubernetes development? Test out [Telepresence for Docker](#) today.

 Search Docker Hub Explore Repositories Organizations Help Upgrade  pengan88

pengan88 Repositories ubuntu General Using 0 of 1 private repositories. [Get more](#)

General Tags Builds Collaborators Webhooks Settings


**Add a short description for this repository**  
The short description is used to index your content on Docker Hub and in search engines. It's visible to users in search results. [Update](#)

 **pengan88 / ubuntu** [Public View](#)

**Description**  
This repository does not have a description [✎](#)  
Last pushed: a few seconds ago

**Docker commands**  
To push a new tag to this repository,  
`docker push pengan88/ubuntu:tagname`

**Tags**  
This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
1.0		Image	---	a few seconds ago

[See all](#) [Go to Advanced Image Management](#)

**Automated Builds**  
Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.  
Available with Pro, Team and Business subscriptions. [Read more about automated builds](#) [↗](#)  
[Upgrade](#)

### 3.4.2.2、Steps to publish the image to tar package:

**\* The following command operate in the host machine, not in docker \***

1、Method 1: docker export and docker import (Export compressed package from container)

Use (docker ps -a) to query the ID of the container and run the following command to generate a compressed package for the container

```
docker export -o xxx.tar [container ID]

#The tar package name can be your own
#Such as: docker export -o ros2-base:1.0.1.tar [container ID]
```

The compressed package is generated and then imported into the image library using docker import

```
docker import xxx.tar [image name]:[tag]

#The image name and tag can be your own
#Such as: docker import ros2-base:1.0.1.tar yahboomtechnology/ros2-base:1.0.1
```

2、Method 2: docker save and docker load (Export compressed package from image)

Use (docker images) to query the ID and tag of the image and run the following command to generate a compressed package for the image

```
docker save -o xxx.tar [image name]:[tag]

#The image name and tag can be your own
#Such as: docker save -o ros2-base:1.0.1.tar yahboomtechnology/ros2-base:1.0.1
```

After generating the compressed package, import it into the image library using docker load

```
docker load -i xxx.tar

# Here is the name of your tar package
```

3、The difference between the two methods

docker export is generated from containers, docker save generated from image. docker export smaller packages than docker save, The reason is that save can save an entire hierarchical file system, export only save container-layer file systems.