3. In-depth understanding of docker images and publishing images

The operating environment and software and hardware reference configuration are as follows:

- Reference model: ROSMASTER X3
- Robot hardware configuration: Arm series main control, Silan A1 lidar, AstraPro Plus depth camera
- Robot system: Ubuntu (no version required) + docker (version 20.10.21 and above)
- PC virtual machine: Ubuntu (18.04) + ROS (Melodic)
- Usage scenario: Use on a relatively clean 2D plane

3.1. Understanding of mirroring

1. An image is a lightweight, executable independent software package that contains everything needed to run a certain software. We package the application and configuration into a formed, deliverable, and deployable operating environment, including code, libraries required for runtime, environment variables, and configuration files, etc. This large packaged operating environment is an image image file.
2. Docker container instances can only be generated through image files.

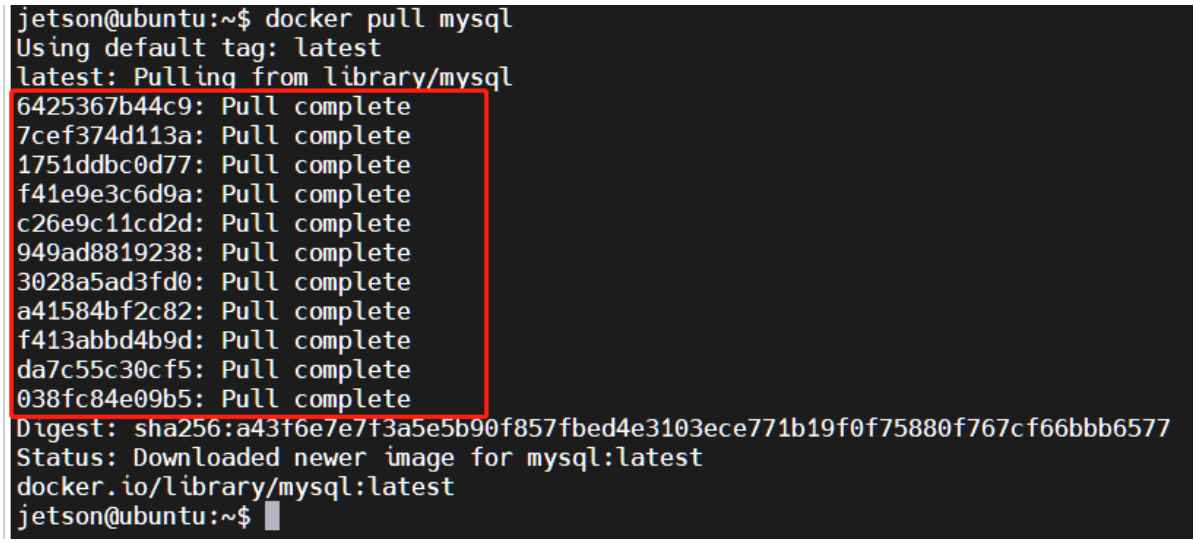
3.2, UnionFS (Union File System)

1. Union File System (UnionFS) is a layered, lightweight, high-performance file system. It is the basis of docker images and supports modifications to the file system to be superimposed layer by layer as a single submission. At the same time, different directories can be mounted under the same virtual file system.
2. Images can be inherited through layering. Based on the basic image, various specific application images can be produced.

Features of the Union file system: multiple file systems are loaded at the same time, but from the outside, only one file system can be seen; Union loading will superimpose the file systems of each layer, so that the final file system will contain all layers. Files and directories.

3.3. Mirror layering

When downloading an image, pay attention to the download log output. You can see that it is downloading layer by layer:




```
# To view the image layering method, you can use 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 basic image layer. When modifications are made or new content is added, a new image layer will be created on top of the current image layer.
- To give a simple example, if a new image is created based on Ubuntu 20.04, this is the first layer of the new image; if a python package is added to the image, a second image layer will be created on top of the basic image layer; If you continue to add a security patch, a third image layer will be created.
- Docker images are read-only, 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 everything below the container is called the mirror layer!

3.3.2. The benefits of layering docker images

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

3.4. Making and publishing images

3.4.1. Make an image

Method 1. Submit an image from the container:

```
# Order
docker commit -m="Description information submitted" -a="Author" Container id
Target image name to be created: [Tag name] [The -m -a parameter can also be
omitted]

# 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
3b9c01839579 hello-world "/hello" 3 hours ago Exited (0) 3 hours ago jovial_brown
jetson@ubuntu:~$ docker commit c54bf9efae47 ubuntu:1.0
sha256:78ca7be949b6412f74ba12e8d16bd548aaa7c3fa25134326db3a67784f848f8f
```

```
jetson@ubuntu:~$ docker images # Generated ubuntu:1.0 image
REPOSITORY TAG IMAGE ID CREATED SIZE
ubuntu 1.0 78ca7be949b6 5 seconds ago 69.2MB
yahboomtechnology/ros-foxy 3.4.0 49581aa78b6b 5 hours ago 24.3GB
yahboomtechnology/ros-foxy 3.3.9 cefb5ac2ca02 4 days ago 20.5GB
yahboomtechnology/ros-foxy 3.3.8 49996806c64a 4 days ago 20.5GB
yahboomtechnology/ros-foxy 3.3.7 8989b8860d17 5 days ago 17.1GB
yahboomtechnology/ros-foxy 3.3.6 326531363d6e 5 days ago 16.1GB
ubuntu latest bab8ce5c00ca 6 weeks ago 69.2MB
hello-world latest 46331d942d63 13 months ago 9.14kB
```

Method 2. Create image using dockerfile:

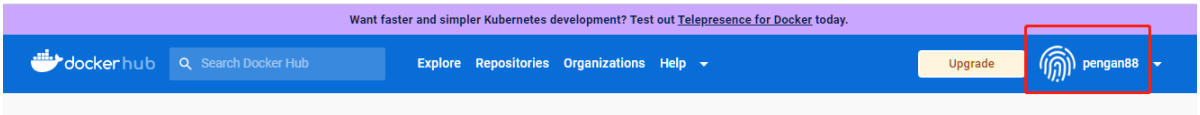
```
# Order
docker build -f dockerfile file path -t new image name: TAG . # There is a . at
the end of the docker build command indicating the current directory
# test
docker build -f dockerfile-ros2 -t yahboomtechnology/ros-foxy:1.2 .

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

3.4.2. Release image

The docker repository is a centralized place for storing image files. The largest public repository is docker hub (https://hub.docker.com/), which stores a large number of images for users to download. Domestic public warehouses include Alibaba Cloud, NetEase Cloud, etc.

Steps to publish the image to docker hub:

1. Address: https://hub.docker.com/, register an account first
2. Ensure that the account can be logged in normally



3. Use the tag command to modify the image name.

The specifications for publishing images to docker hub are:

```
docker push registered user name/image name
```

For example, my registered user name here is: pengan88, then I need to change the image name first.

```
# Order:
docker tag image ID modified image name

# 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 78ca7be949b6pengan88/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、登录docker hub发布镜像：

```
jetson@ubuntu:~$ docker login -u pengan88
Password:      # 这里输入docker hub注册的账号密码
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、访问docker hub可查看到已经发布成功