1.Overview and installation

Docker official website: http://www.docker.com

Docker Chinese website: https://www.docker-cn.com

Docker Hub (warehouse) official website: https://hub.docker.com

Foreword:

Starting from this chapter, you will enter Muto's ros2 course. All ros2 courses are placed in docker containers. Customers can experience and learn the use of containerized development methods

When learning the ros2 course, there are two points to note:

1. Please turn off the WiFi hotspot that comes with Muto, connect Muto to your own WiFi network, and ensure that it can connect to the external network, which will facilitate subsequent learning

2. Please turn off the self-starting APP control program at startup, otherwise it will occupy the device and cause unpredictable errors. For operation tutorials, please refer to "03. Remote control experience\1. Open and close the APP control program"

The above two points of attention will be throughout the entire course of ros2, please be sure to pay attention.

1.0. Recommended software tools for operating docker on Windows

It is recommended to use the MobaXterm remote tool to connect to the muto master

For the installation and use of MobaXterm, please refer to the link below or Baidu.

```
https://www.jb51.net/program/285577y15.htm
```

1.1.Docker overview

Docker is an application container engine project, developed based on Go language and open source.

1.1.1.Why docker appears?

Let me first mention a few scenarios:

1. The operation and maintenance team will deploy the project developed for you to the server and tell you that there is a problem and it cannot be started. You ran it locally and found that there is no problem...
2. The project to be launched is unavailable due to some software version updates...
3. Some projects involve a lot of environmental content, including various middlewares, various configurations, and the deployment of many servers...

These problems are actually related to the environment.
To avoid various problems caused by different environments, it is best to deploy the project together with the various environments required by the project.
For example, if the project involves redis, mysql, jdk, es and other environments, bring the entire environment with you when deploying the jar package. So the question is, how can we bring the environment with the project?

Docker is here to solve this problem!

1.1.2. Docker’s core idea

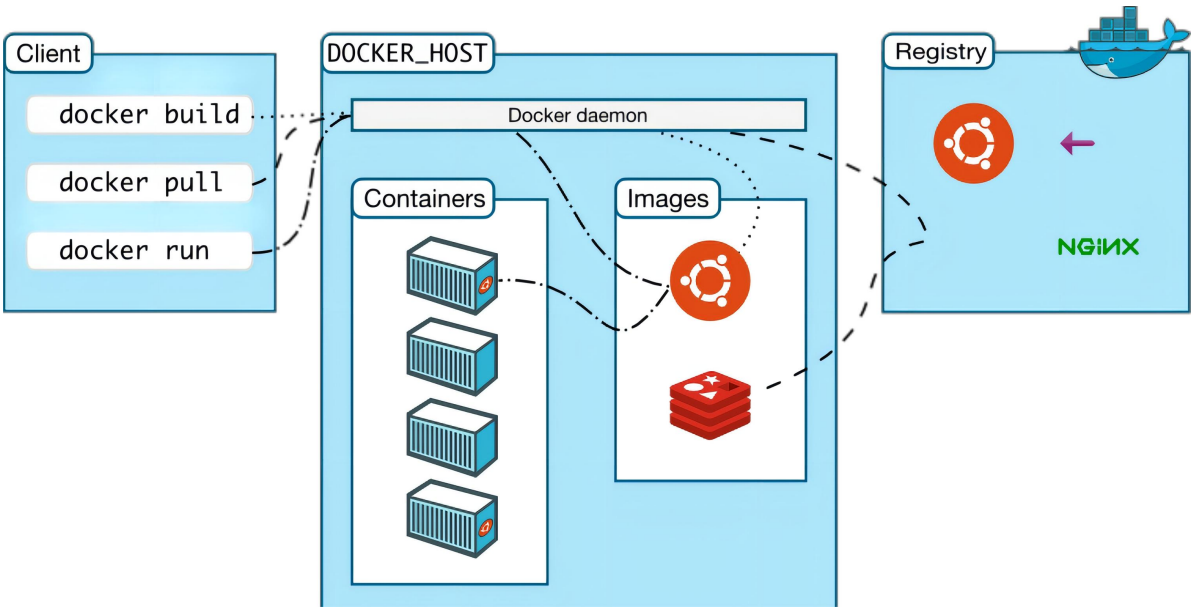
This is the logo of Docker, a whale full of containers. On the back of the whale, the containers are isolated from each other. This is the core idea of Docker.
For example, if multiple applications were running on the same server before, there may be conflicts in the software's port occupancy. Now, after isolation, they can run independently. In addition, docker can maximize the use of server capabilities.

1.1.3. Comparison between virtual machine and Docker

- The docker daemon can communicate directly with the main operating system to allocate resources to each docker container; it can also isolate the container from the main operating system and isolate each container from each other. A virtual machine takes minutes to start, while a docker container can start in milliseconds. Since there is no bloated slave operating system, docker can save a lot of disk space and other system resources.
- Virtual machines are better at completely isolating the entire operating environment. For example, cloud service providers often use virtual machine technology to isolate different users. Docker is usually used to isolate different applications, such as front-end, back-end and database.
- Docker containers are more resource-saving and faster than virtual machines (starting, shutting down, creating, and deleting)

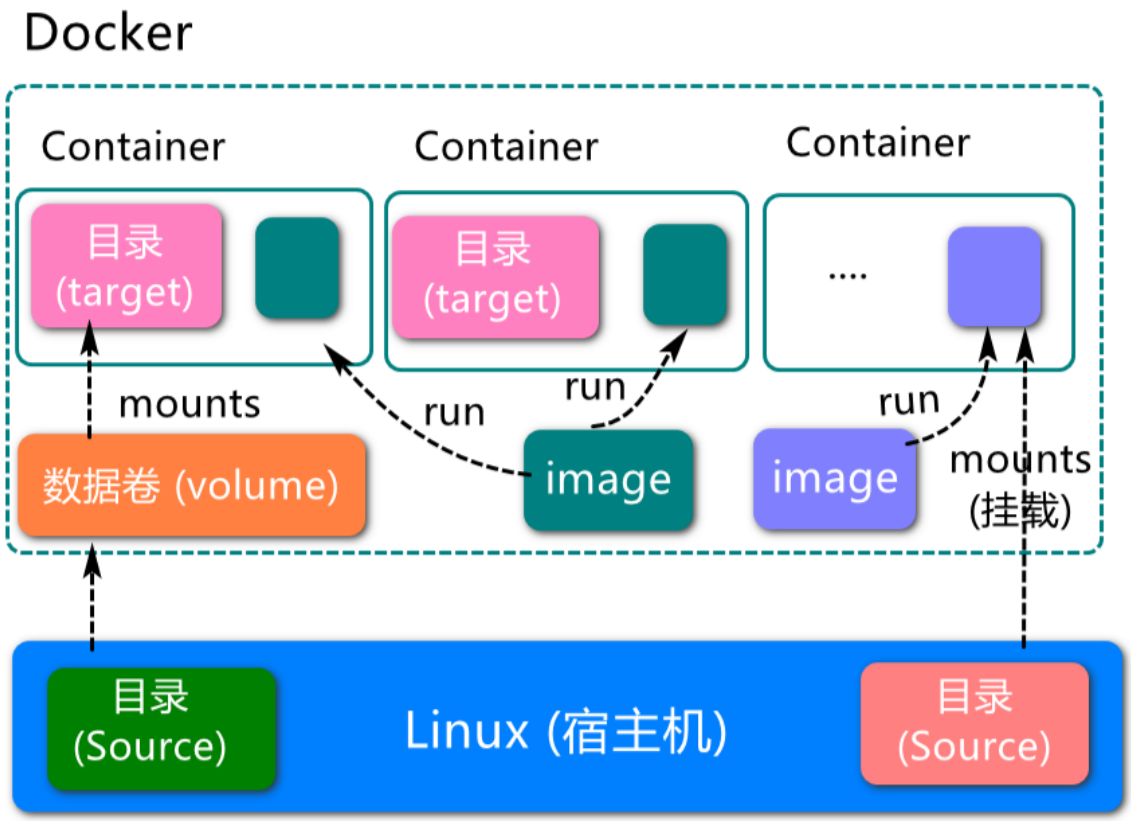
1.1.4. Docker architecture

docker uses client-server architecture. The docker client communicates with the docker daemon, which is responsible for the heavy lifting of building, running, and distributing docker containers. The docker client and daemon can run on the same system, or the docker client can be connected to a remote docker daemon. The docker client and daemon communicate using REST API through UNIX sockets or network interfaces. Another docker client is docker compose, which allows you to work with applications composed of a set of containers.



- docker client is the docker command used directly after installing docker.
- docker host is our docker host (that is, the operating system with docker installed)
- Docker daemon is the background daemon process of docker, which listens and processes docker client commands and manages docker objects such as images, containers, networks and volumes.
- Registry is a remote warehouse where Docker pulls images, providing a large number of images for download. After the download is completed, they are saved in images (local image warehouse).
- images is docker's local image warehouse. You can view image files through docker images.

1.1.5. Docker core objects



1.1.6.Image, Container, Repository:

Image:

```
The docker image (Image) is a read-only template. Images can be used to create
docker containers, and one image can create many containers. Just like classes
and objects in Java, classes are images and containers are objects.
```

Container:

```
Docker uses a container to run an application or a group of applications
independently. A container is a running instance created using an image. It can
be started, started, stopped, deleted. Each container is isolated from each other
to ensure a secure platform. A container can be thought of as a simplified
version of the Linux environment (including root user permissions, process space,
user space, network space, etc.) and the applications running in it. The
definition of a container is almost exactly the same as that of an image. It is
also a unified perspective of a bunch of layers. The only difference is that the
top layer of the container is readable and writable.
```

Repository:

```
A repository is a place where image files are stored centrally. Warehouses are
divided into two forms: public warehouses (public) and private warehouses
(private).
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.
```

It is necessary to correctly understand the concepts of Repository/Image/Container:

- Docker itself is a container running carrier or management engine. We package the application and configuration dependencies to form a deliverable running environment. This packaged running environment is like an image file. Only through this image file can a docker container be generated. The image file can be thought of as a template for the container. docker generates an instance of the container based on the image file. The same image file can generate multiple container instances running simultaneously.
- The container instance generated by the image file is itself a file, called an image file.
- A container runs a service. When we need it, we can create a corresponding running instance through the docker client, which is our container.
- As for the Repository, it is a place where a bunch of images are placed. We can publish the images to the warehouse and pull them out of the warehouse when needed.

1.1.7.Docker operating mechanism

docker pull execution process:

1. The client sends instructions to the docker daemon
2. The docker daemon first checks whether there are related images in the local images.
3. If there is no relevant mirror locally, request the mirror server to download the remote mirror to the local

docker run execution process:

1. Check whether the specified image exists locally. If it does not exist, download it from the public warehouse.
2. Create and start a container using the image
3. Allocate a file system (simplified Linux system) and mount a read-write layer outside the read-only mirror layer
4. Bridge a virtual interface from the bridge interface configured on the host to the container.
5. Configure an IP address from the address pool to the container
6. Execute user-specified applications

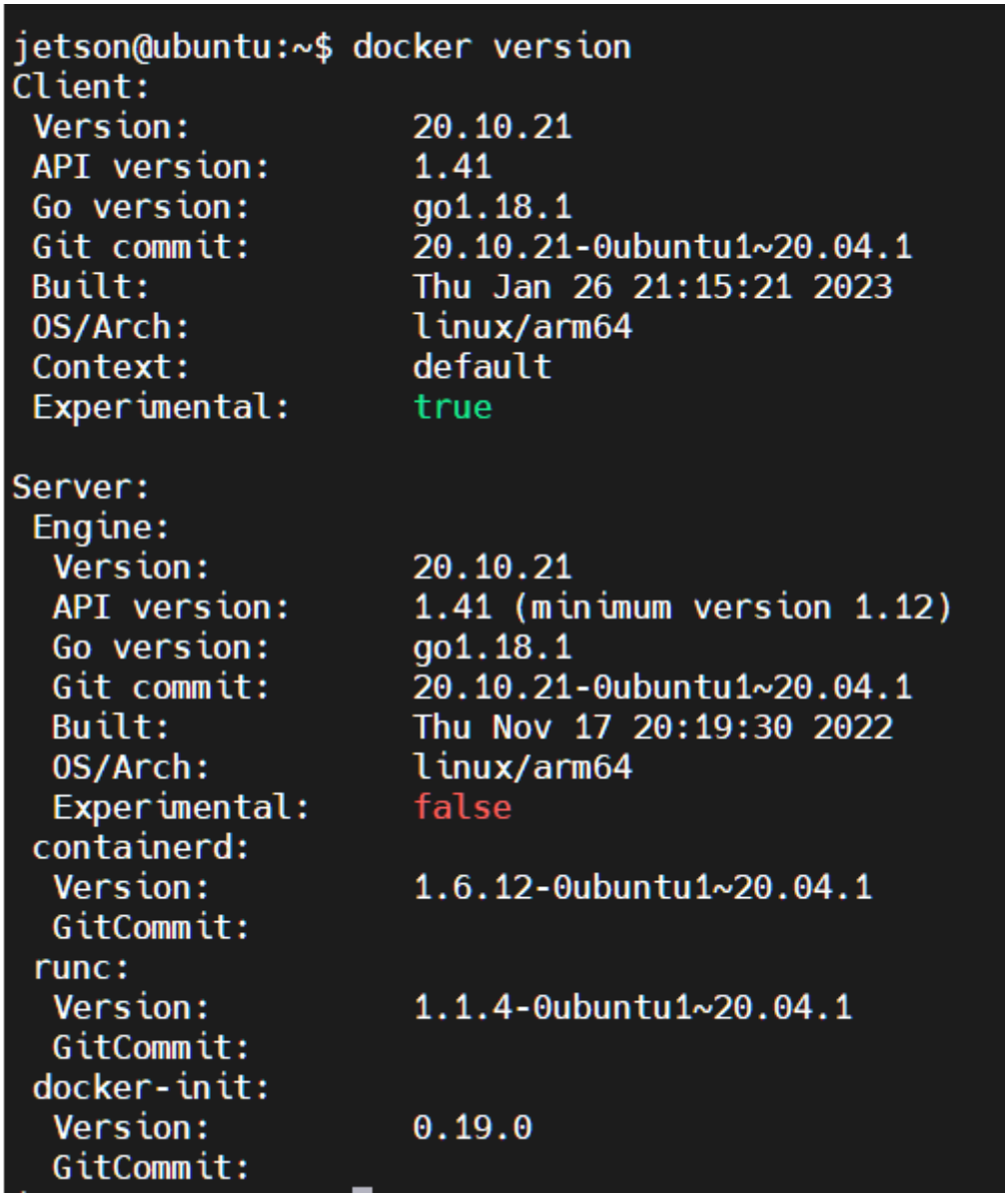
1.2. Docker installation

1. Official website installation reference manual: https://docs.docker.com/engine/install/ubuntu/
2. You can use the following command to install it with one click:

```
curl -fsSL https://get.docker.com | bash -s docker --mirror Aliyun
```

3. View the docker version

```
sudo docker version
```



4. Test command

```
sudo docker run hello-world
```

The following output indicates that the Docker installation is successful.

```
jetson@ubuntu:~$ sudo docker run hello-world
[sudo] password for jetson:
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
7050e35b49f5: Pull complete
Digest: sha256:4e83453afed1b4fa1a3500525091dbfca6ce1e66903fd4c01ff015dbcb1ba33e
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (arm64v8)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```