1. Docker overview and Docker installation

***ORIN's ROS1 course is all in the Docker container. ***

1. Docker overview and Docker installation

- 1.1. Docker overview
  - 1.1.1, Why does docker appear
  - 1.1.2. The core idea of Docker
  - 1.1.3. Comparison between virtual machines and Docker
  - 1.1.4, Docker architecture
  - 1.1.5、Docker core objects
  - 1.1.6、Image, container, warehouse
  - 1.1.7, Docker operation mechanism
- 1.2, docker 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

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

- Reference vehicle: ROSMASTER X3PLUS
- Robot hardware configuration: Arm series controller, EAI 4ROS LiDAR, AstraPro Plus depth camera
- Robot system: Ubuntu (no version requirement) + Docker (20.10.21 and above)
- PC virtual machine: Ubuntu (20.04) + ROS (Noetic)
- Usage scenario: Use on a relatively clean 2D plane

1.1. Docker overview

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

1.1.1, Why does docker appear

First, let me mention a few scenarios:

1. The operation and maintenance team deploys the project you developed to the server and tells you that there is a problem and it cannot be started. You run it locally and find that there is no problem...
2. The project to be launched is unavailable due to the update of some software versions...
3. Some projects involve a lot of environment content, various middleware, various configurations, and many servers need to be deployed...

In fact, these problems are 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.

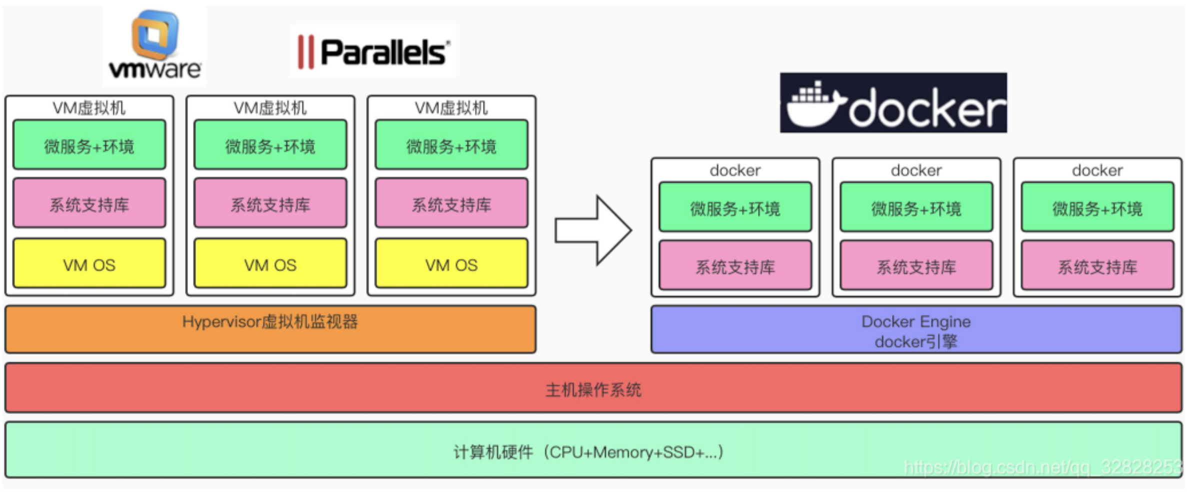
For example, the project involves redis, mysql, jdk, es and other environments. When deploying the jar package, bring the entire environment. Then the question is, how can the project bring the environment together?

Docker is here to solve this problem!

1.1.2. The core idea of Docker

This is the Docker logo, a whale full of containers. On the whale's back, 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 might be conflicts in the port usage of the software. Now they can run independently after isolation. In addition, Docker can maximize the use of the server's capabilities.

1.1.3. Comparison between virtual machines and Docker

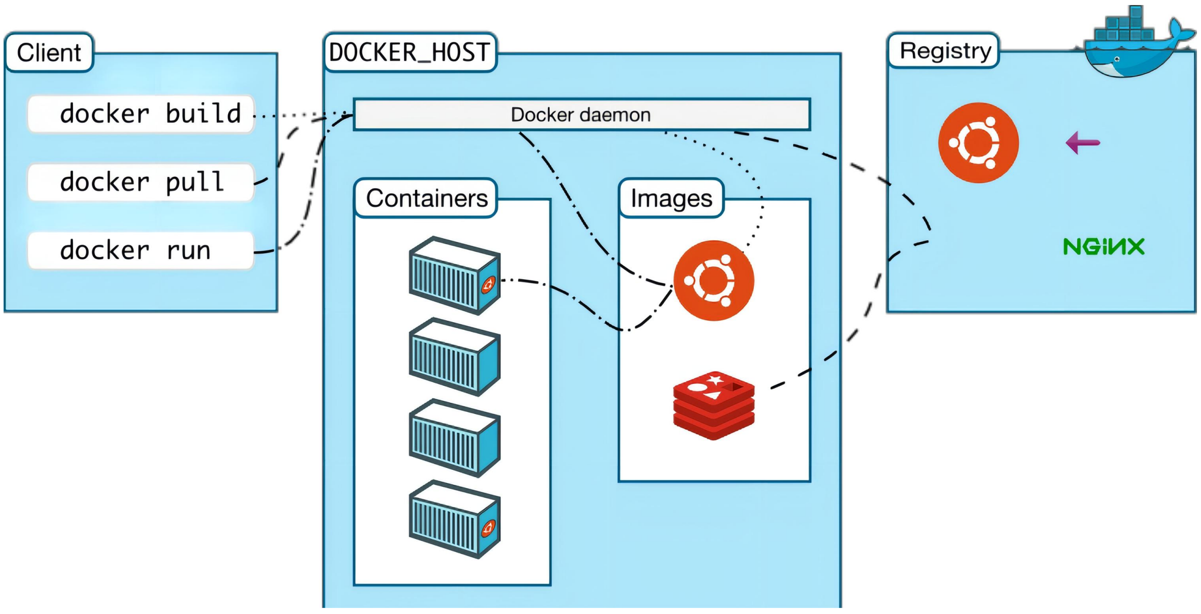


- The docker daemon can communicate directly with the main operating system to allocate resources to each docker container; it can also isolate containers from the main operating system and isolate each container from each other. It takes minutes to start a virtual machine, while a docker container can 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-efficient and faster than virtual machines (starting, shutting down, creating, deleting)

1.1.4, Docker architecture

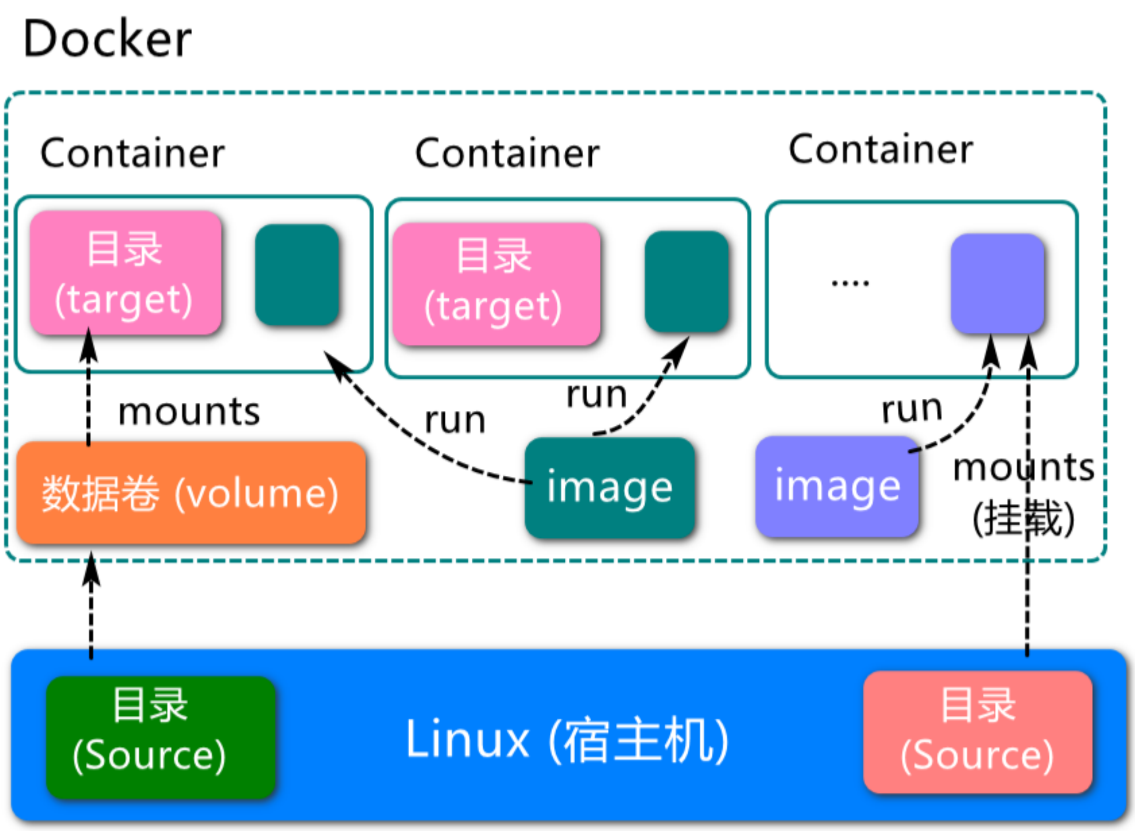
Docker uses a client-server architecture. The Docker client communicates with the Docker daemon, which is responsible for building, running, and distributing Docker containers. The Docker client and daemon can run on the same system, or you can connect the Docker client to a remote Docker daemon. The Docker client and daemon communicate using the REST API over a

UNIX socket or a network interface. Another Docker client is Docker Compose, which allows you to handle applications consisting 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 of Docker, which listens and processes Docker client commands and manages Docker objects, such as images, containers, networks, and volumes.
- Registry is the remote warehouse where Docker pulls images, providing a large number of images for download. After downloading, they are saved in images (local image warehouse).
- Images is the local image warehouse of Docker, and image files can be viewed through Docker images.

1.1.5、Docker core objects



1.1.6、Image, container, warehouse

Image:

```
A docker image is a read-only template. An image can be used to create a docker
container, and one image can create many containers. Just like classes and
objects in Java, classes are images, and containers are objects.
```

Container:

```
Docker uses containers to run one or a group of applications independently. A
container is a running instance created using an image. It can be started,
started, stopped, and deleted. Each container is isolated from each other to
ensure a secure platform. You can think of a container as a simplified version of
a Linux environment (including root user permissions, process space, user space,
and network space, etc.) and the applications running in it. The definition of a
container is almost exactly the same as an image, and it is also a unified
perspective of a stack 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. Repositories are
divided into public and 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
repositories include Alibaba Cloud and NetEase Cloud.
```

It is necessary to correctly understand the concepts of storage/image/container:

- Docker itself is a container operation carrier or management engine. We package the application and configuration dependencies to form a deliverable operating environment, which is like an image image file. Only through this image file can a docker container be generated. The image file can be regarded 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 also 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 warehouse, it is a place where a bunch of images are placed. We can publish the image to the warehouse and pull it down from the warehouse when needed.

1.1.7, Docker operation mechanism

Docker pull execution process:

1. The client sends the command to the docker daemon
2. The docker daemon first checks whether there are relevant images in the local images
3. If there is no relevant image locally, it requests the image server to download the remote image to the local

Docker run execution process:

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

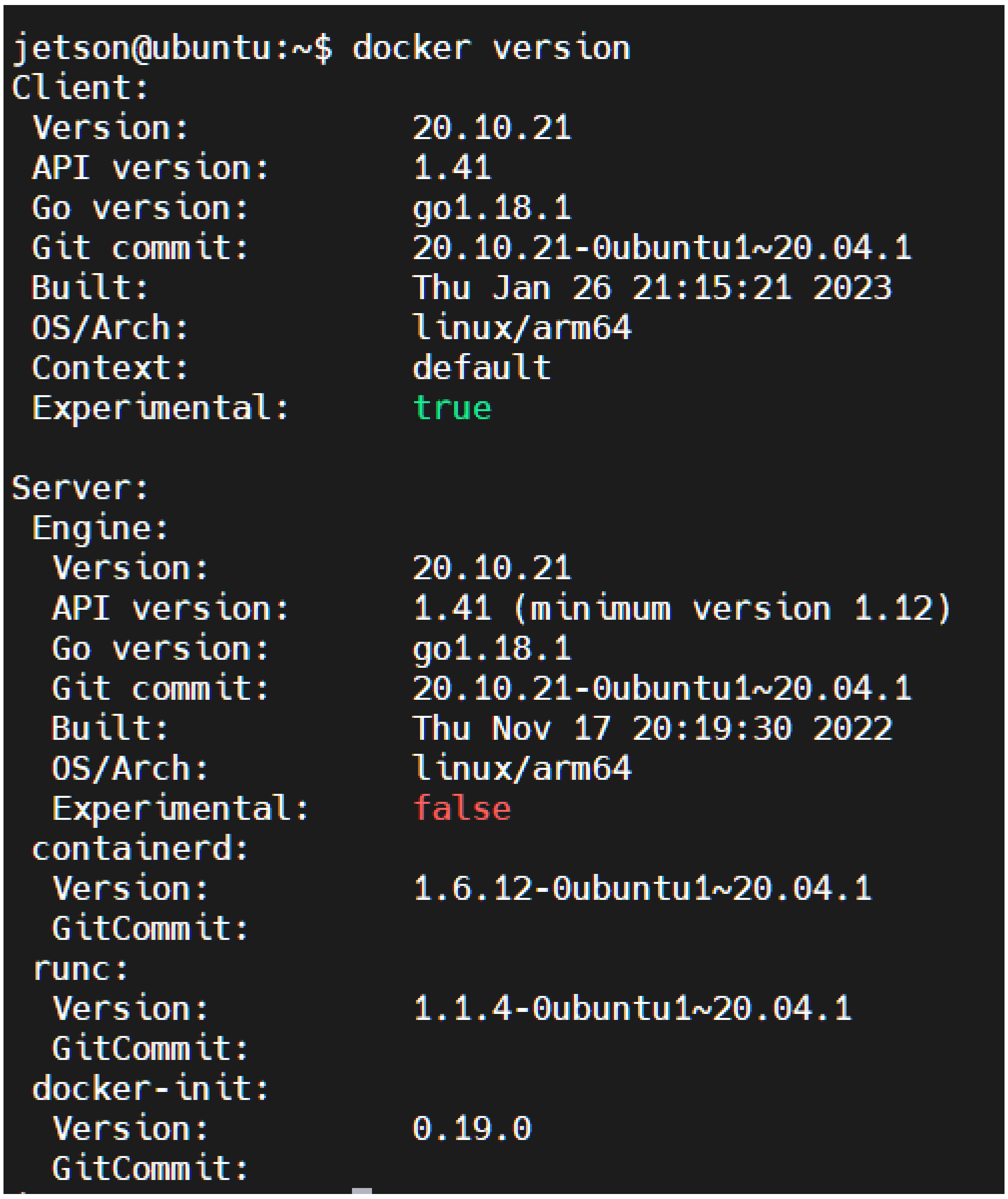
1.2, docker installation

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

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

3. Check the docker version

```
sudo docker version
```



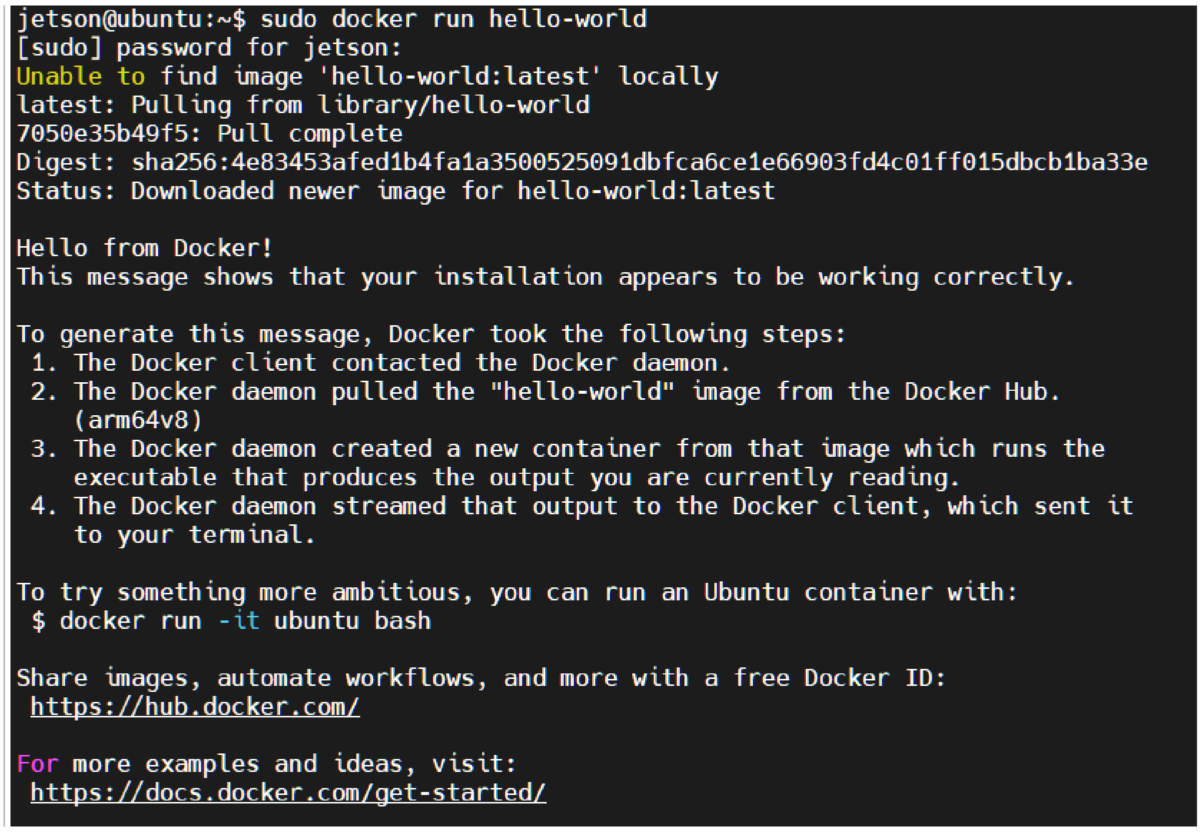
```
jetson@ubuntu:~$ docker version
Client:
 Version:           20.10.21
 API version:       1.41
 Go version:        go1.18.1
 Git commit:        20.10.21-0ubuntu1~20.04.1
 Built:             Thu Jan 26 21:15:21 2023
 OS/Arch:           linux/arm64
 Context:           default
 Experimental:      true

Server:
 Engine:
  Version:          20.10.21
  API version:      1.41 (minimum version 1.12)
  Go version:       go1.18.1
  Git commit:       20.10.21-0ubuntu1~20.04.1
  Built:            Thu Nov 17 20:19:30 2022
  OS/Arch:          linux/arm64
  Experimental:     false
 containerd:
  Version:          1.6.12-0ubuntu1~20.04.1
  GitCommit:
 runc:
  Version:          1.1.4-0ubuntu1~20.04.1
  GitCommit:
 docker-init:
  Version:          0.19.0
  GitCommit:
```

4. Test command

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

The following output indicates that Docker is successfully installed



```
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/
```