Docker Introduction and Installation

Docker Official Website: http://www.docker.com

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

Docker Hub (Repository) Official Website: https://hub.docker.com

1.1. Overview of Docker

Docker is an open-source application container engine developed in the Go language.

1.1.1 Why Docker Happens

Let's consider a few scenarios:

1. Operations and maintenance help you deploy your project to a server and tell you there's a problem and it won't start. You run it locally and find no problems...
2. A project you're about to launch becomes unavailable due to a software update...
3. Some projects involve a wide range of environments, including various middleware, configurations, and multiple servers...

In summary, these issues all stem from the environment. To avoid issues caused by different environments, it's best to deploy all the necessary environments along with your project when deploying it. For example, if your project requires Redis, MySQL, JDK, ES, and other environments, deploying the JAR package should include all of these environments. The question then becomes, how can you ensure your project includes all of these environments?

Docker is the solution!

1.1.2 The Core Concept of Docker

This is the Docker logo: a whale filled with containers. On its back, the containers are isolated from each other. This is the core concept of Docker. For example, if multiple applications were previously running on the same server, there might be port conflicts. Now, with isolation, they can run independently. Furthermore, Docker maximizes the server's capabilities.

1.1.3 Comparing Virtual Machines and Docker

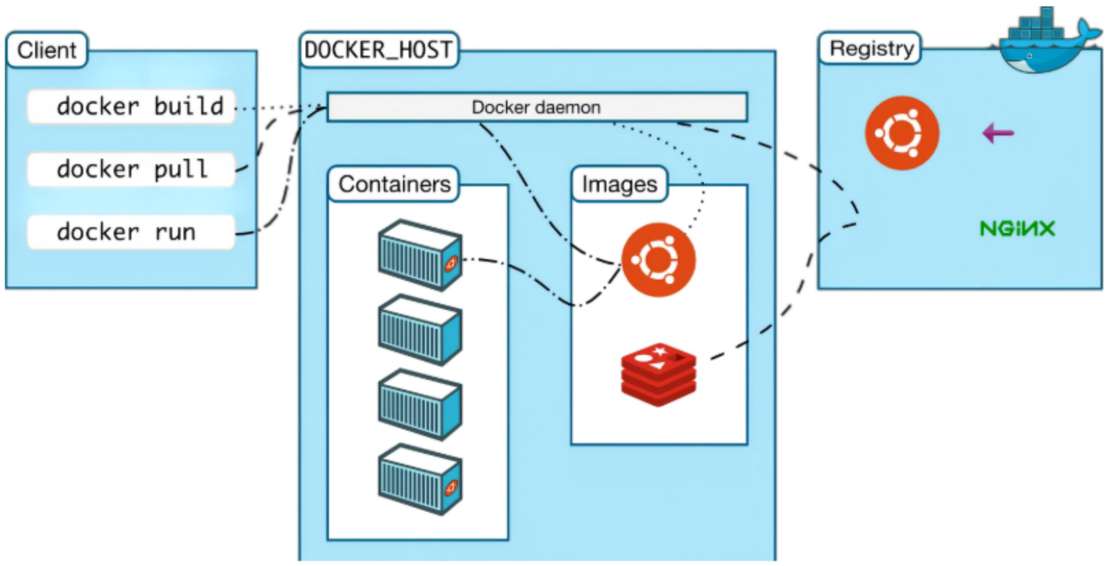
- The Docker daemon communicates directly with the main operating system to allocate resources for each Docker container. It also isolates containers from the main operating system and from each other. While virtual machines take minutes to boot, Docker containers can boot in milliseconds. Without a bloated secondary operating system, Docker can save significant disk space and other system resources.
- Virtual machines are better at completely isolating entire operating environments. For example, cloud service providers often use virtual machines to isolate different users.

Docker, on the other hand, is often used to isolate different applications, such as front-ends, back-ends, and databases.

- Docker containers are more resource-efficient and faster (starting, shutting down, creating, and deleting) than virtual machines.

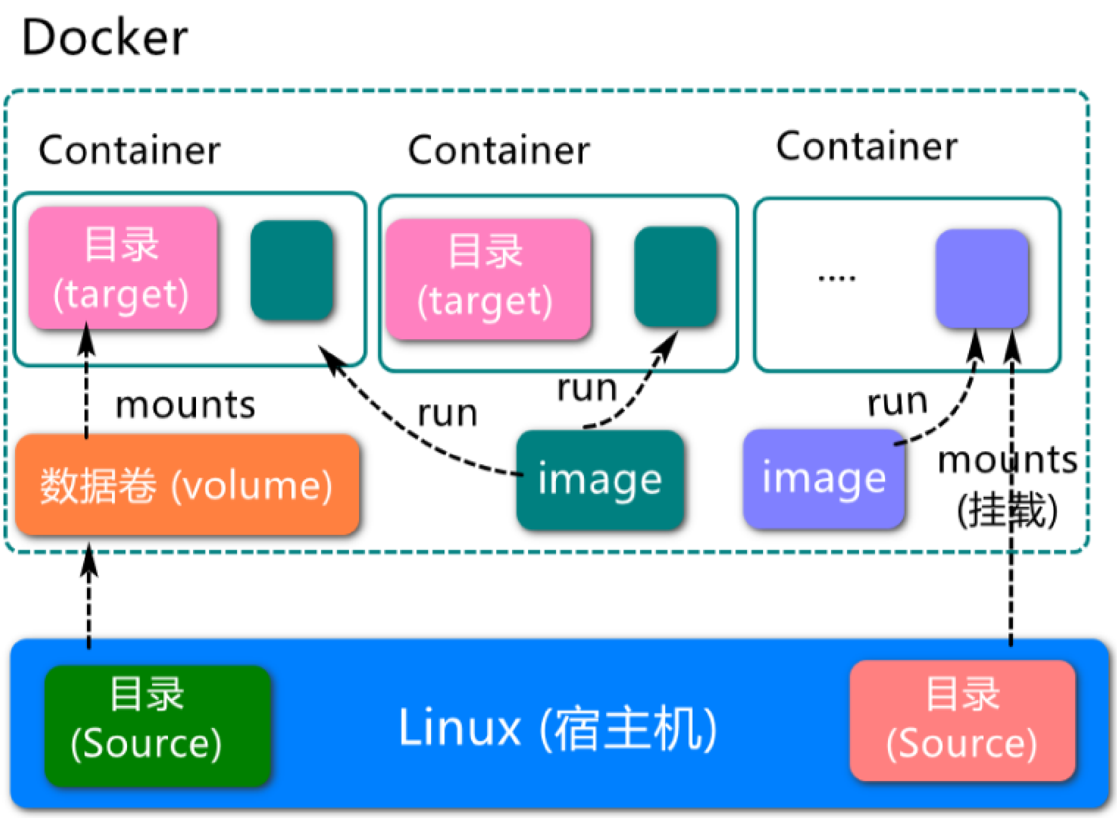
1.1.4 Docker Architecture

Docker uses a client-server architecture. Docker clients communicate with the Docker daemon, which handles 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 connect to a remote Docker daemon. The Docker client and daemon communicate using a REST API over a UNIX socket or a network interface. Another Docker client is Docker Compose, which allows you to manage applications composed of a set of containers.



- The Docker client is the Docker command used directly after installing Docker.
- The Docker host is the Docker host (the operating system on which Docker is installed).
- The Docker daemon is the Docker background daemon process that listens for and processes Docker client commands and manages Docker objects such as images, containers, networks, and volumes.
- The registry is a remote repository from which Docker pulls images. It provides a large number of images for download and stores them in the "images" (local image repository).
- The "images" directory is the local Docker image repository, where you can view image files.

1.1.5. Docker Core Objects



1.1.6. Images, Containers, and Repositories

Images:

```
A Docker image is a read-only template. Images can be used to create Docker
containers, and one image can create many containers. It's like classes and
objects in Java: a class is an image, and a container is an object.
```

Containers:

```
Docker uses containers to independently run one or a group of applications. A
container is a running instance created from an image. It can be started,
started, stopped, and deleted. Each container is isolated from the others,
ensuring a secure platform. Think of a container as a simplified Linux
environment (including root user permissions, process space, user space, and
network space) and the applications running within it. The definition of a
container is almost identical to an image, representing a unified view of a stack
of layers. The only difference is that the topmost layer of a container is
readable and writable.
```

Repository:

```
A repository is a centralized location for storing image files. Repositories are
categorized as public or private.
The largest public repository is Docker Hub (https://hub.docker.com/), which
houses a vast number of images for download. Domestic public repositories include
Alibaba Cloud and NetEase Cloud.
```

It's important to correctly understand the concepts of repository, image, and container:

- Docker itself is a container runtime or management engine. We package applications and configuration dependencies into a deliverable runtime environment, which is like an image file. Only with this image file can a Docker container be generated.An image file can be considered a container template. Docker generates container instances 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 needed, we can create a corresponding running instance, our container, using the Docker client.
- A repository is a place where a bunch of images are stored. We can publish images to the repository and pull them down when needed.

1.1.7. Docker Operation Mechanism

Docker pull execution process:

1. The client sends a command to the Docker daemon.
2. The Docker daemon first checks whether the relevant image exists in its local images.
3. If the relevant image does not exist locally, it requests the image server to download the remote image locally.

Docker run execution process:

1. Checks whether the specified image exists locally. If not, it downloads it from a public repository.
2. Creates and starts a container using the image.
3. Allocates a file system (a simplified Linux system) and mounts a read-write layer outside the read-only image layer.
4. Bridges a virtual interface from the host's configured bridge interface to the container.
5. Configures an IP address for the container from the address pool.
6. Executes the user-specified application.

1.2. Docker Installation

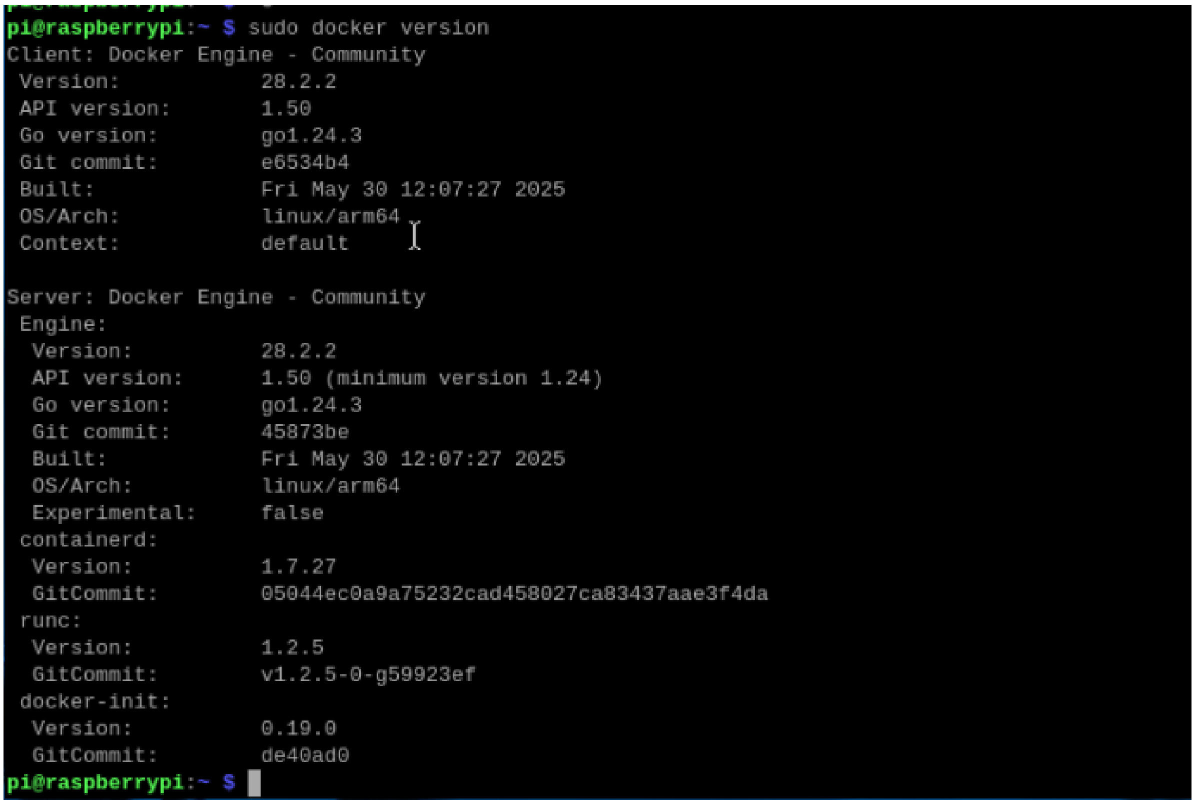
1. Official Installation Reference: https://docs.docker.com/engine/install/ubuntu/
2. One-click installation can be performed using the following command:

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

⚠If network issues such as 443 occur during the installation process, you will need to use a reliable internet connection.

3. Check the Docker version

```
sudo docker version
```



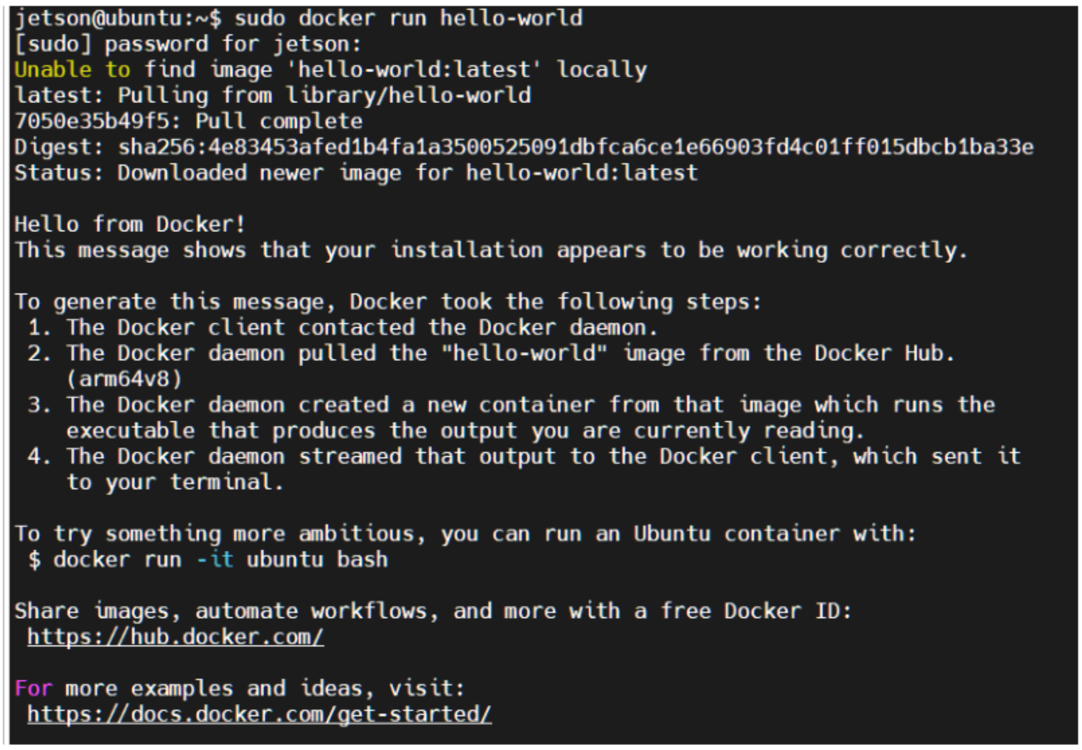
```
pi@raspberrypi:~ $ sudo docker version
Client: Docker Engine - Community
 Version:           28.2.2
 API version:       1.50
 Go version:        go1.24.3
 Git commit:        e6534b4
 Built:             Fri May 30 12:07:27 2025
 OS/Arch:           linux/arm64
 Context:           default

Server: Docker Engine - Community
 Engine:
  Version:          28.2.2
  API version:      1.50 (minimum version 1.24)
  Go version:       go1.24.3
  Git commit:       45873be
  Built:            Fri May 30 12:07:27 2025
  OS/Arch:          linux/arm64
  Experimental:     false
 containerd:
  Version:          1.7.27
  GitCommit:        05044ec0a9a75232cad458027ca83437aae3f4da
 runc:
  Version:          1.2.5
  GitCommit:        v1.2.5-0-g59923ef
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
pi@raspberrypi:~ $
```

4. Test the command

⚠If network issues such as 443 occur during the installation process, you will need to use a reliable internet connection.

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

The following output indicates that Docker has been 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/
```