

1. Overview and installation

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

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

1.1 Docker overview

Docker is an application container engine project, developed based on the Go language, 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 unusable 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. So the question is, how can we bring the project and the environment together?

Docker is here to solve this problem!

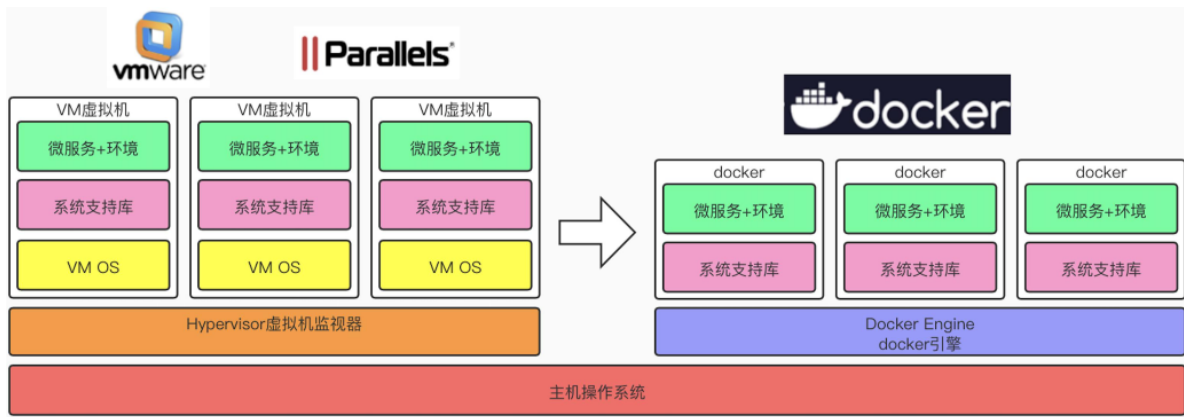
1.1.2 the core idea of Docker



This is the docker logo, a whale full of containers. On the back of the whale, the containers are isolated from each other, which 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.

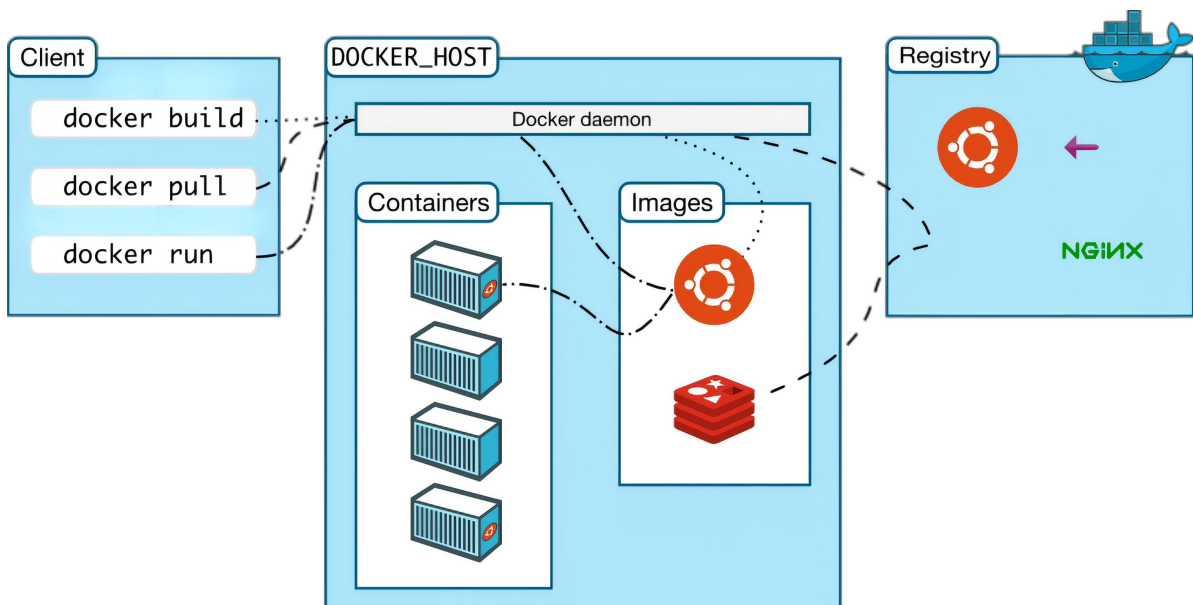
1.1.3. Comparison between virtual machines and Docker



- 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 (start, shut down, create, delete) than virtual machines

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.



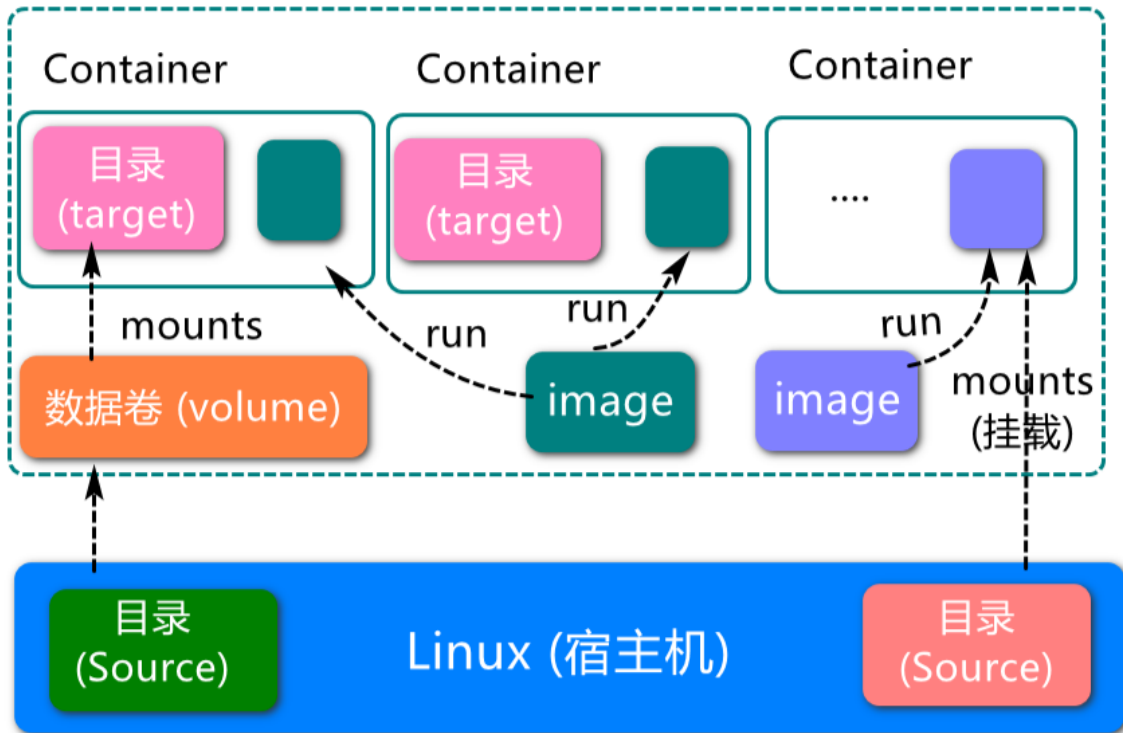
- Docker client is the Docker command used directly after installing Docker.
- Docker host is our Docker host (that is, the operating system with Docker installed)
- Docker daemon is the background daemon of Docker, which listens to and processes Docker client commands and manages Docker objects, such 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

Docker



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 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 at the same time.
- 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 images to the warehouse and pull them 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 are no relevant images 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 (simplified Linux system) and mount a readable and writable 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

Note: Docker is installed at the factory and does not need to be installed again.

Installation reference manual: [Install Docker Engine on Debian](#) | [Docker Docs](#)