上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

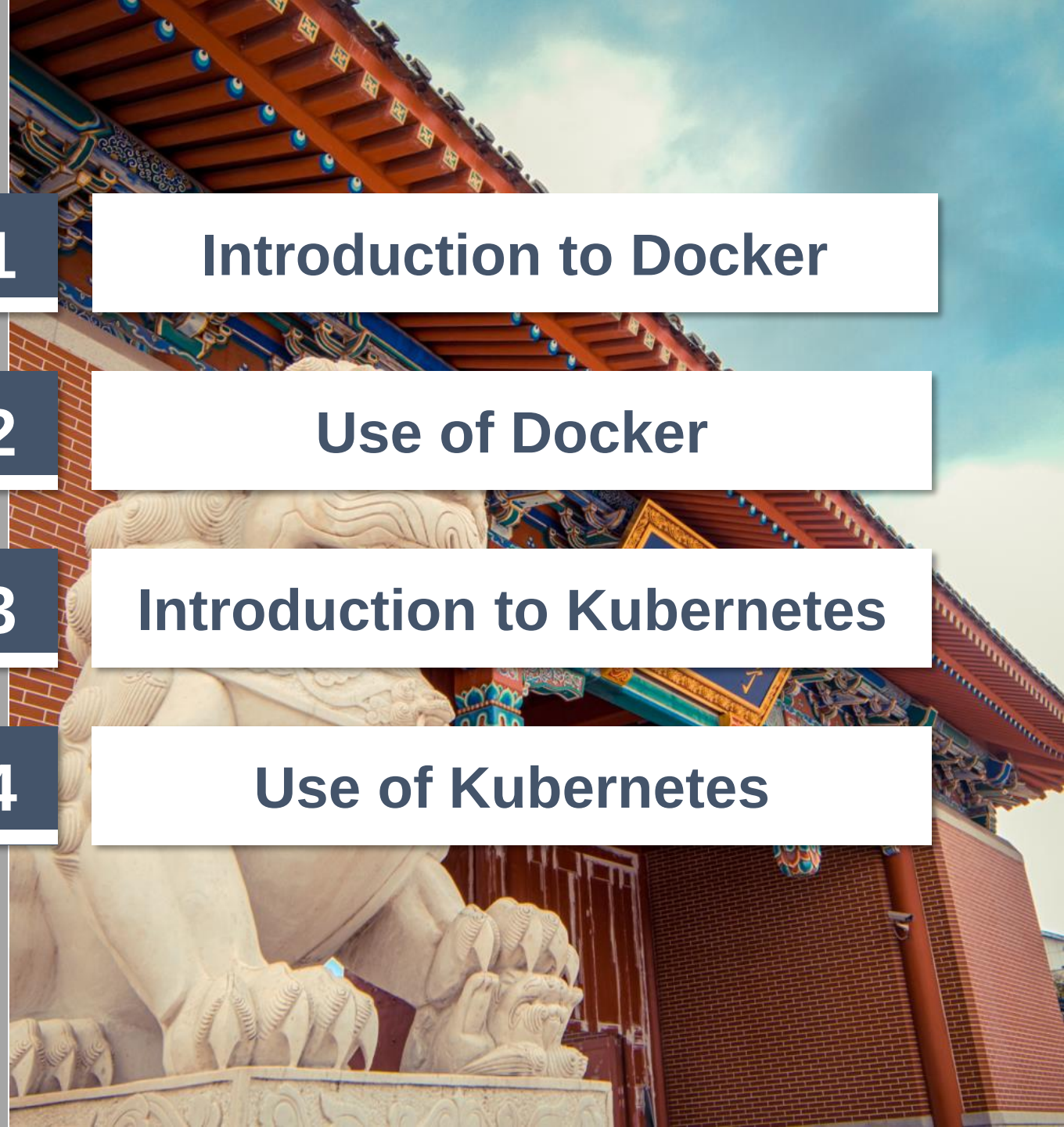
SJTU

轻量级虚拟化：以容器为例

马汝辉 副教授、博导
计算机科学与工程系
上海交通大学

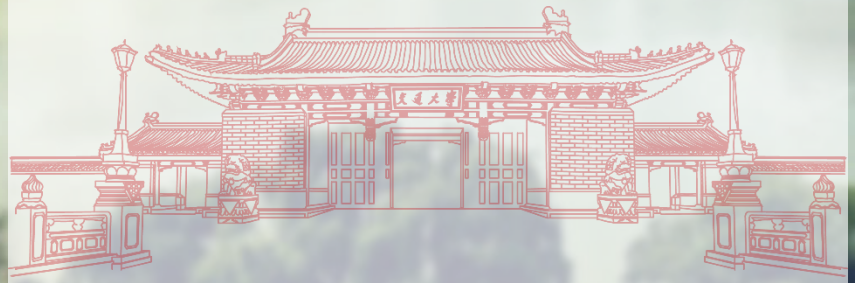
上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

目录



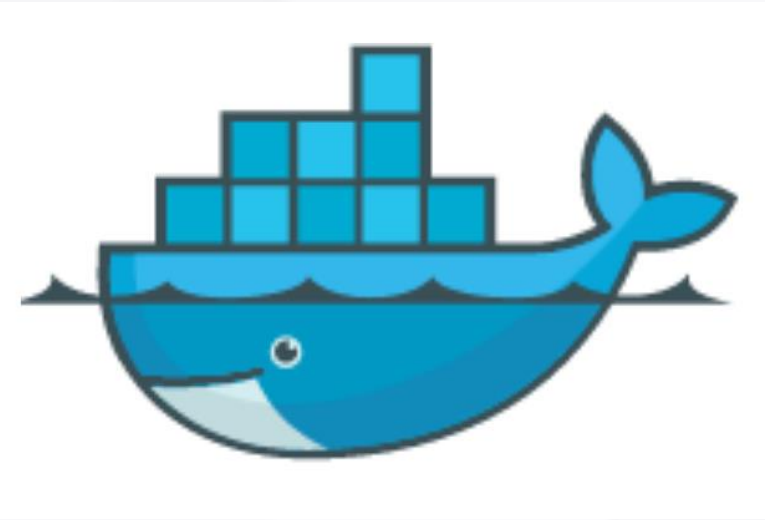
01

Introduction to Docker



Overview

Docker is an open platform for developing, shipping, and running applications, an open source application container engine, which is based on *Go* language and complies with *apache2.0* protocol.

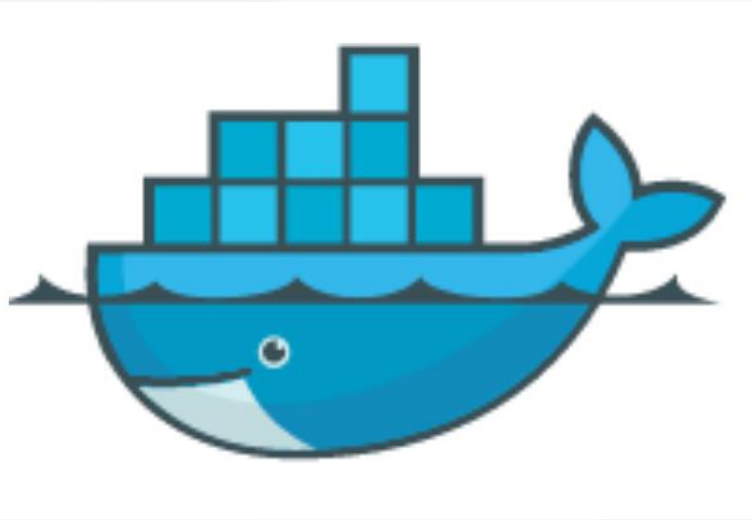


Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications.

By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

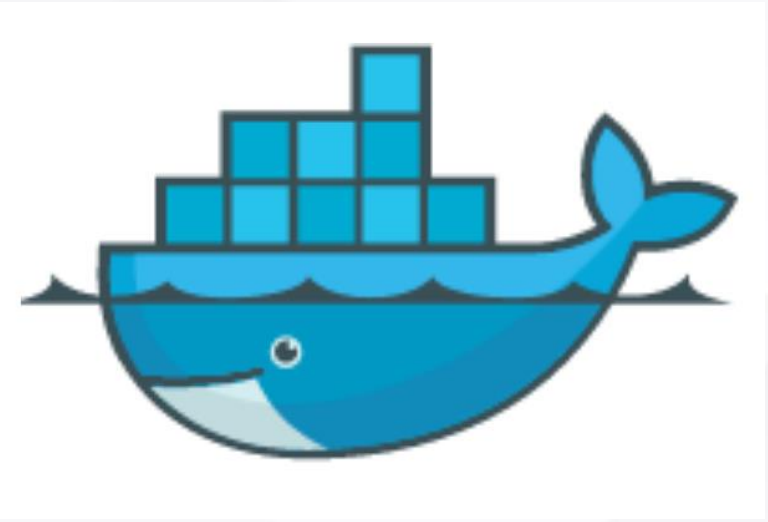
Application scenario

- Automatic packaging and publishing of applications.
- Application isolation.
- Automated testing and continuous integration, release.
- Deploying and adjusting databases or other background applications in a service-oriented environment.
- Compiling from scratch or extending the existing OpenShift or Cloud Foundry platform to build your own PAAS environment.



Advantages

(1) Fast, consistent delivery of your applications
Docker streamlines the development lifecycle by allowing developers to work in standardized environments using local containers which provide your applications and services.



Containers are great for continuous integration and continuous delivery (CI/CD) workflows.

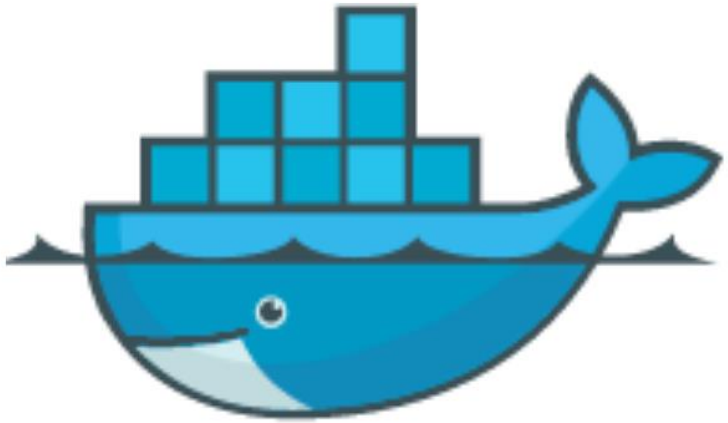
(2) Responsive deployment and scaling
Docker’s container-based platform allows for highly portable workloads. Docker containers can run on a developer’s local laptop, on physical or virtual machines in a data center, on cloud providers, or in a mixture of environments.

Docker’s portability and lightweight nature also make it easy to dynamically manage workloads, scaling up or tearing down applications and services as business needs dictate, in near real time.

Advantages

(3) Running more workloads on the same hardware
Docker is lightweight and fast. It provides a viable, cost-effective alternative to hypervisor-based virtual machines, so you can use more of your compute capacity to achieve your business goals.

Docker is perfect for high density environments and for small and medium deployments where you need to do more with fewer resources.

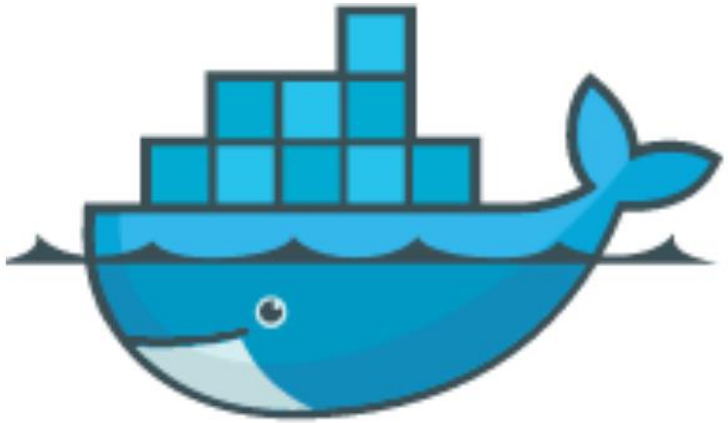


The Docker platform

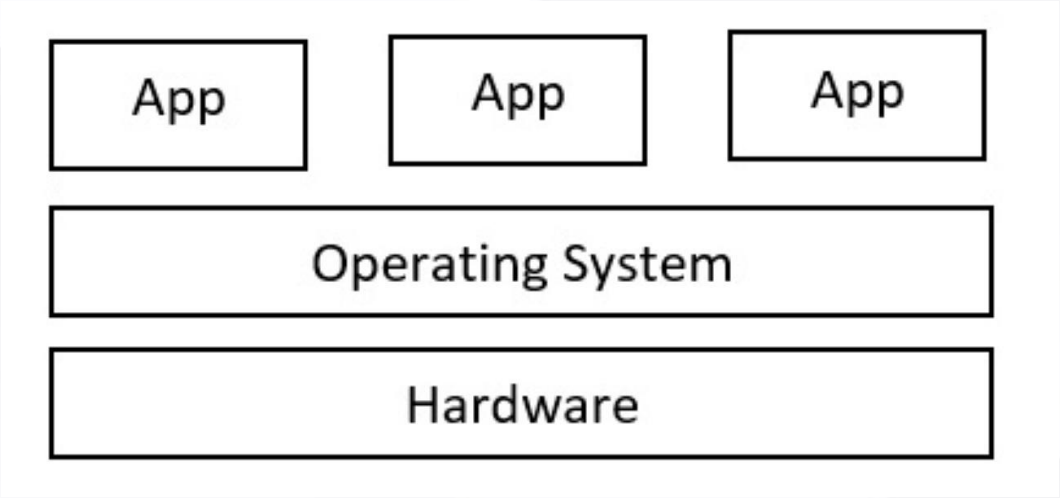
- Docker provides the ability to package and run an application in a loosely isolated environment called a container. The isolation and security allows you to run many containers simultaneously on a given host.

- Containers are lightweight and contain everything needed to run the application, so you do not need to rely on what is currently installed on the host. You can easily share containers while you work, and be sure that everyone you share with gets the same container that works in the same way.

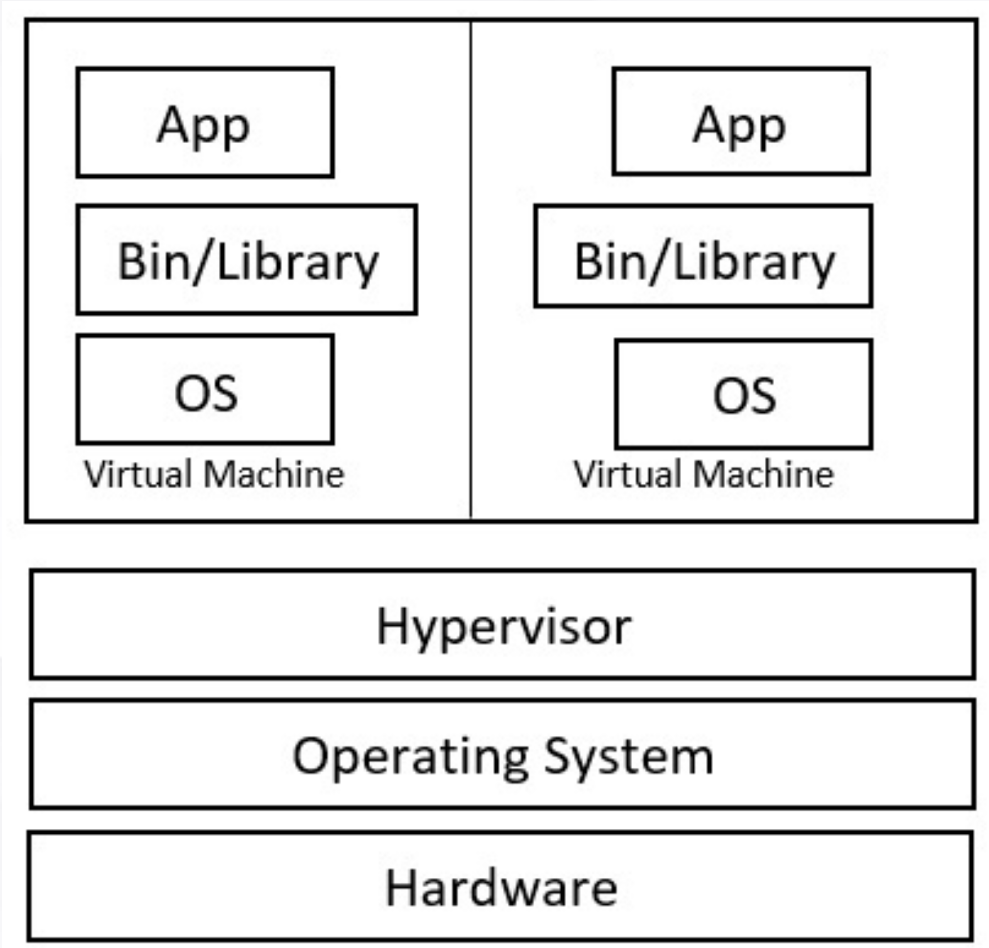
- Docker provides tooling and a platform to manage the lifecycle of your containers. You can develop your application and its supporting components using containers.The container becomes the unit for distributing and testing your application.



Container vs Virtual Machine

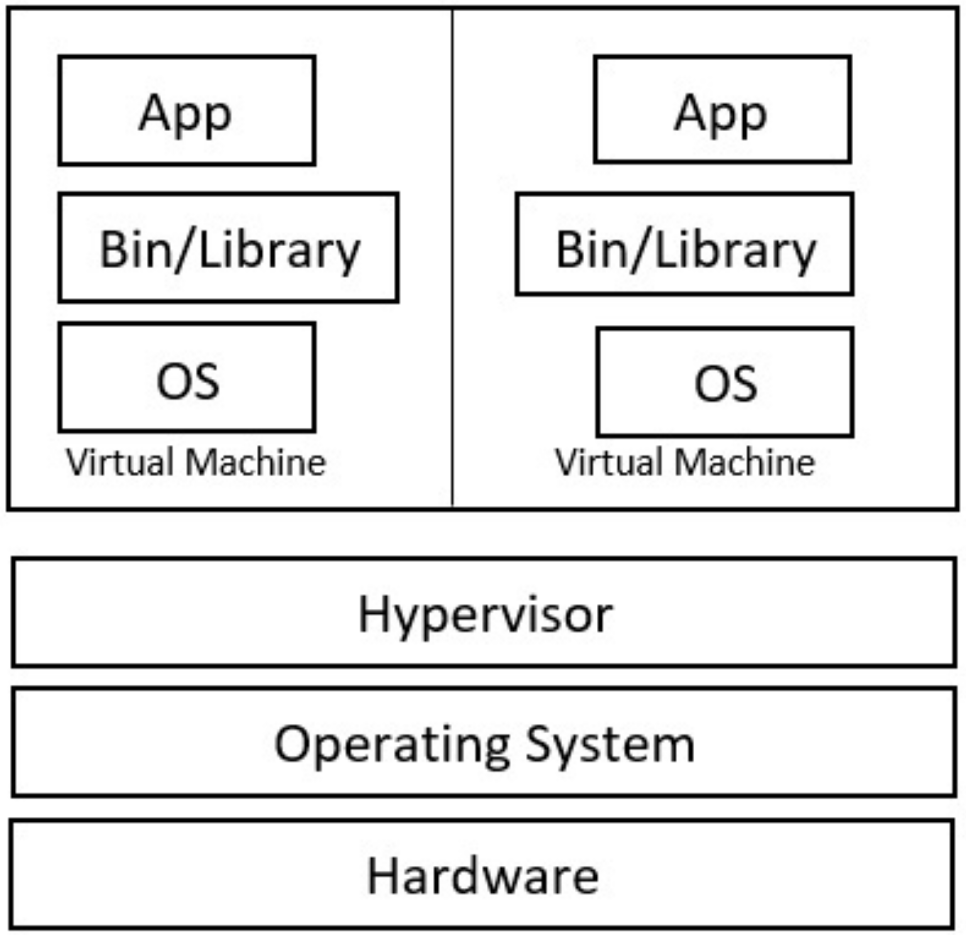


phyical server

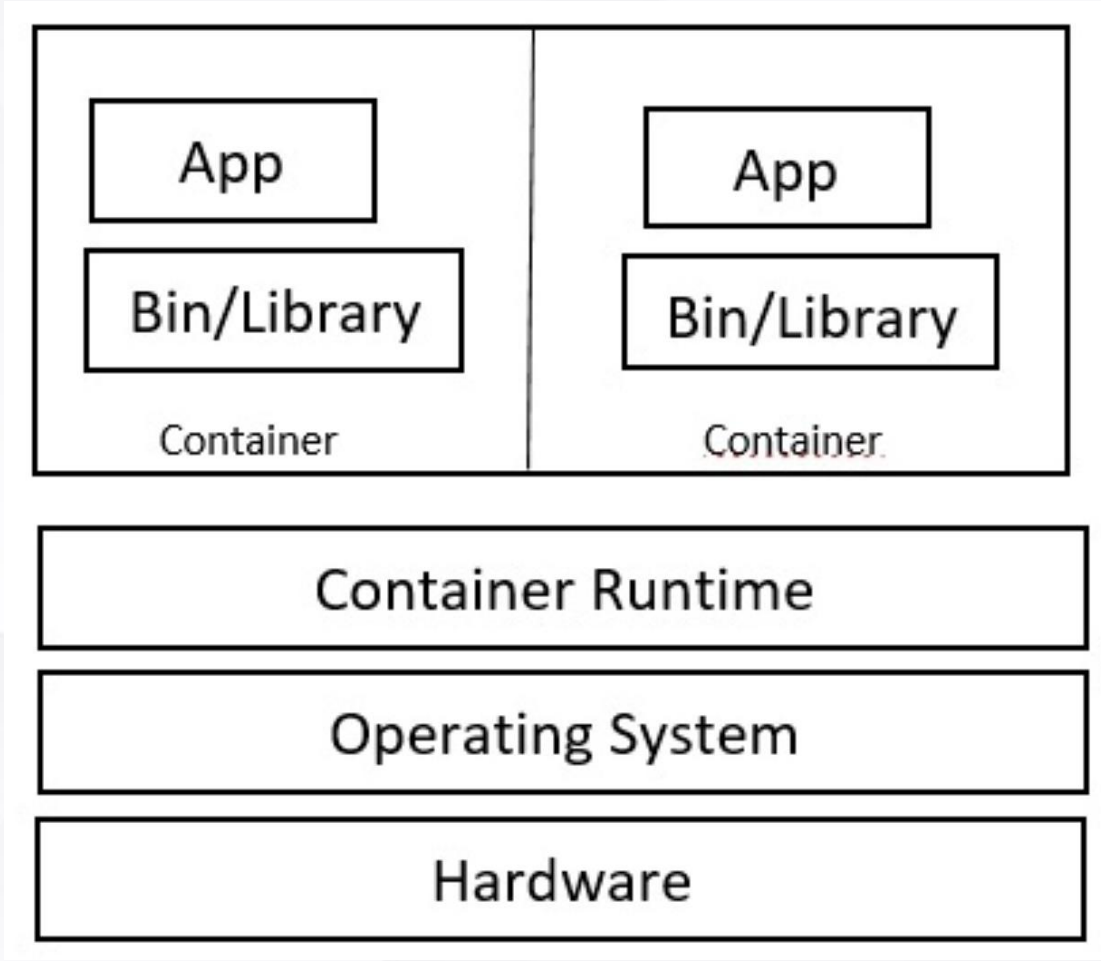


virtual machine

Container vs Virtual Machine



virtual machine



container

Container vs Virtual Machine

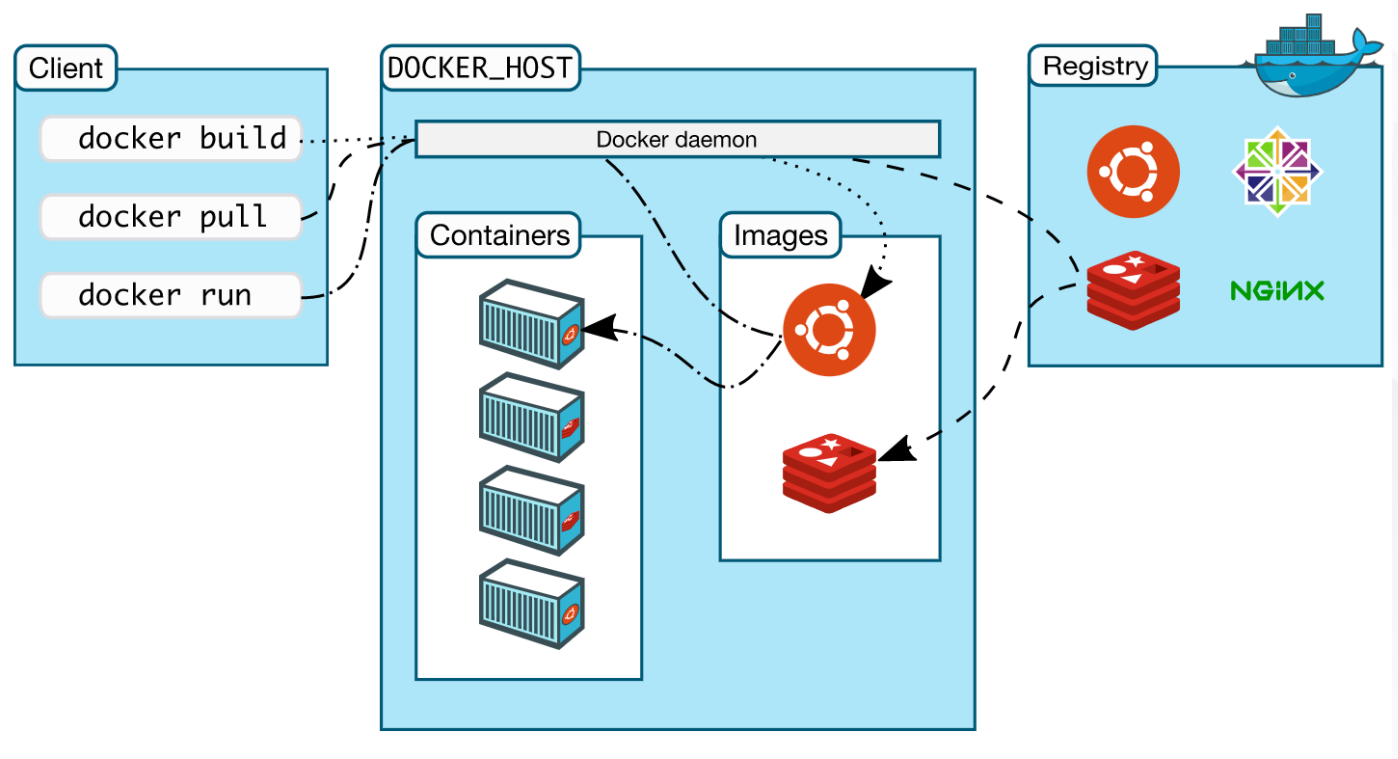
Characteristics	Docker	Virtual Machine
starting speed	seconds level	minutes level
shipping/deployment	consistent development, testing and production environment	-
performance	close to physical machine	large performance loss
image size	KB ~ MB	GB
migration/extention	cross platform replicable	-

Architecture

Docker uses a **client-server(C/S) architecture**.

The Docker client talks to the Docker daemon, which does the heavy lifting of building, running, and distributing your Docker containers. The Docker client and daemon can run on the same system, or you can connect a Docker client to a remote Docker daemon.

The Docker client and daemon communicate using a REST API, over UNIX sockets or a network interface. (https://docs.docker.com/develop/sdk/)



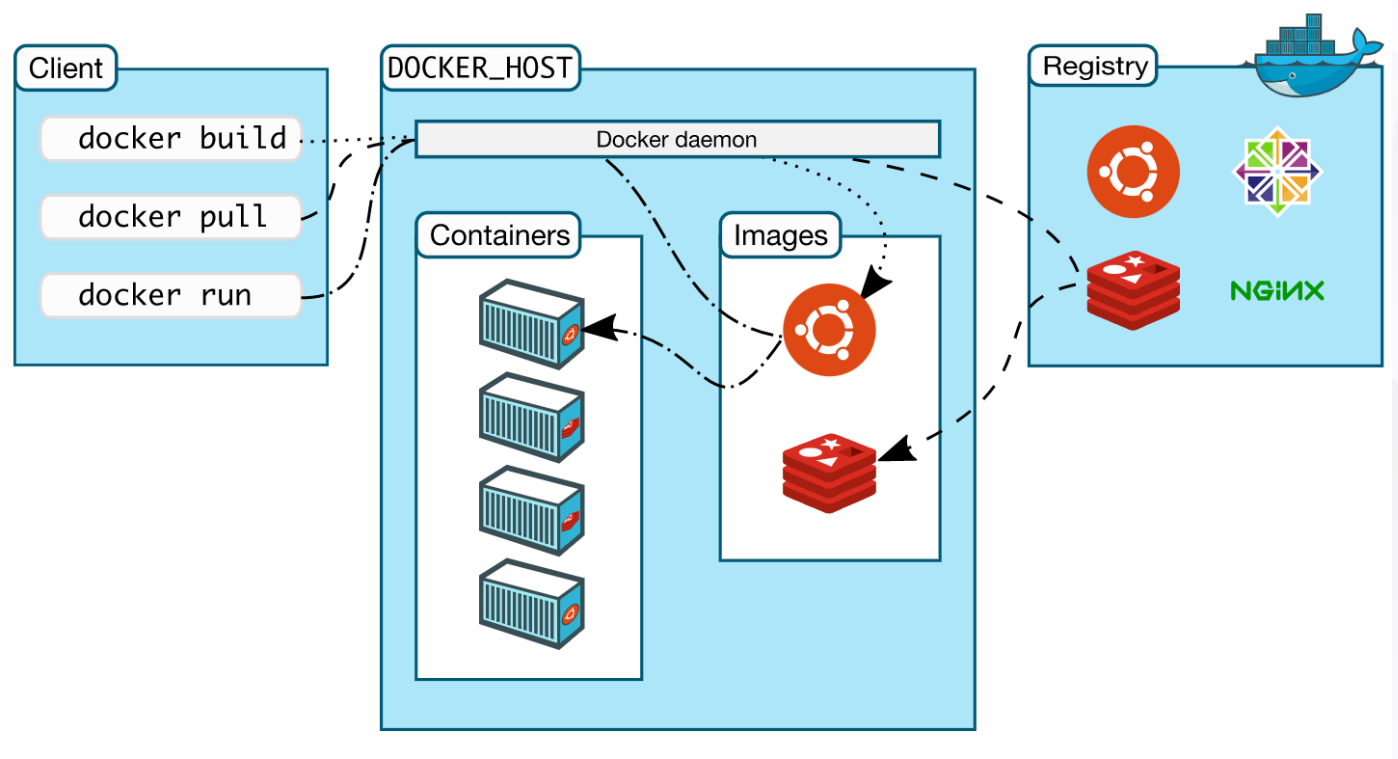
Architecture

(1) The Docker daemon

The Docker daemon (*dockerd*) listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes. A daemon can also communicate with other daemons to manage Docker services.

(2) The Docker client

The Docker client (*docker*) is the primary way that many Docker users interact with Docker. When you use commands such as *docker run*, the client sends these commands to *dockerd*, which carries them out. The *docker* command uses the Docker API. The Docker client can communicate with more than one daemon.

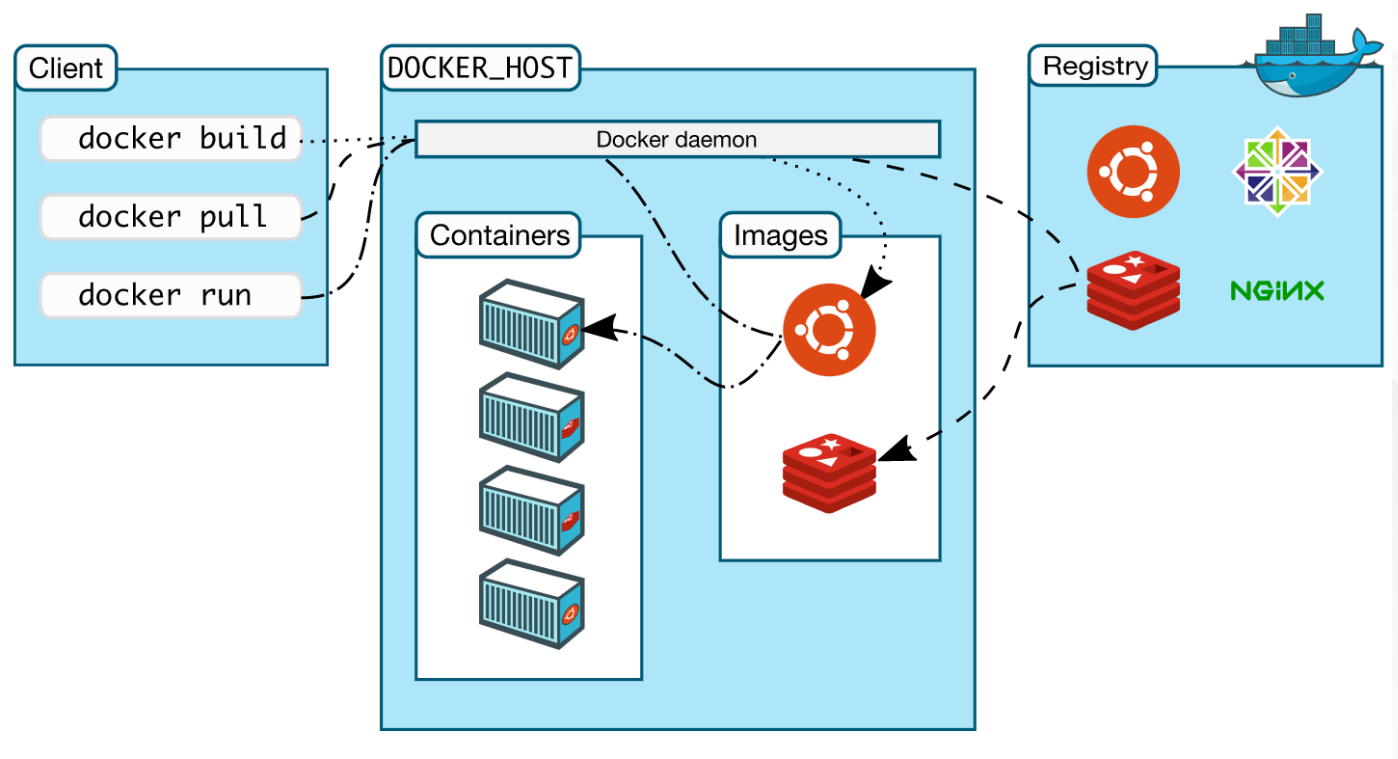


Architecture

(3) Docker registries

A Docker registry stores Docker images. Docker Hub is a public registry that anyone can use, and Docker is configured to look for images on Docker Hub by default. You can even run your own private registry.

When you use the *docker pull* or *docker run* commands, the required images are pulled from your configured registry. When you use the *docker push* command, your image is pushed to your configured registry.

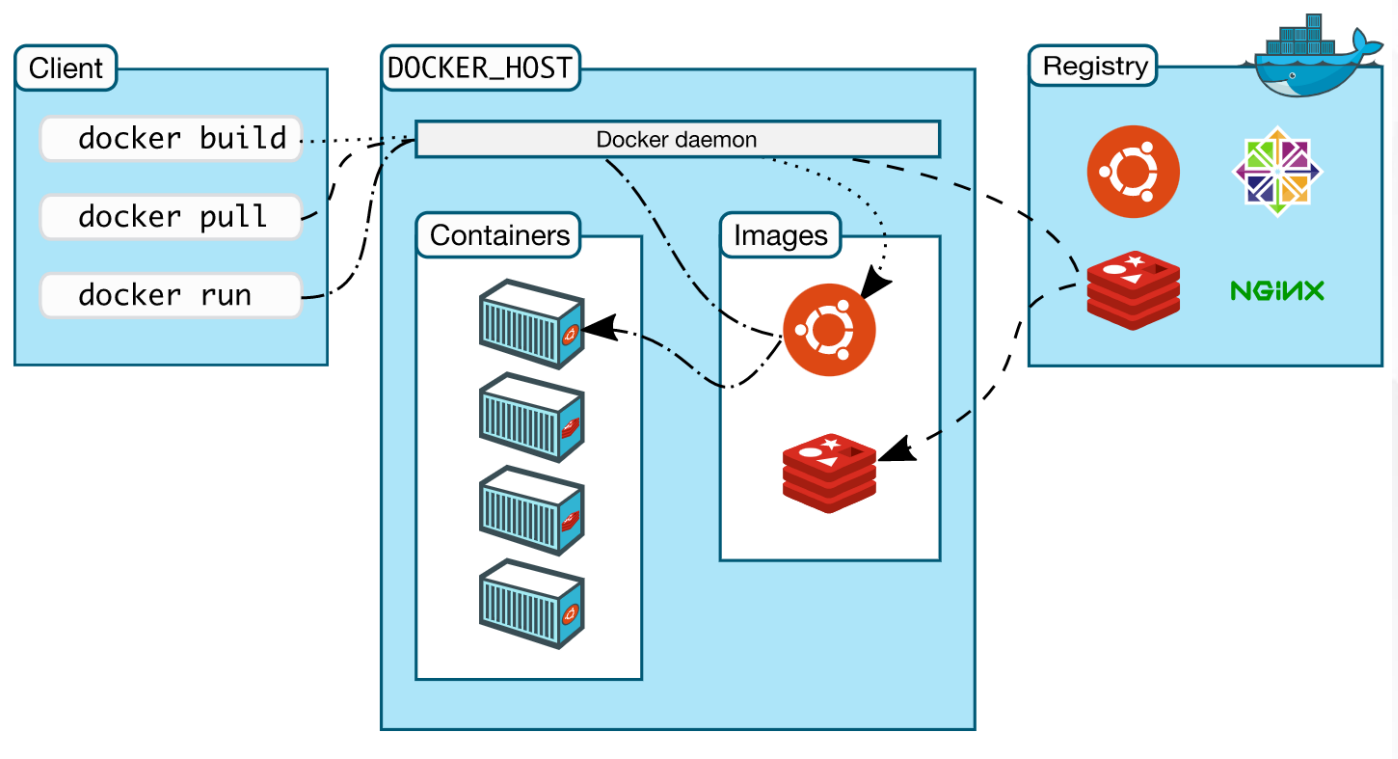


Architecture

(4) Docker objects —— Images

An image is a read-only template with instructions for creating a Docker container.

Often, an image is based on another image, with some additional customization.You might create your own images or you might only use those created by others and published in a registry. To build your own image, you create a *Dockerfile* for defining the steps needed to create the image and run it. Each instruction in a Dockerfile creates a layer in the image. When you change the Dockerfile and rebuild the image, only those layers which have changed are rebuilt.



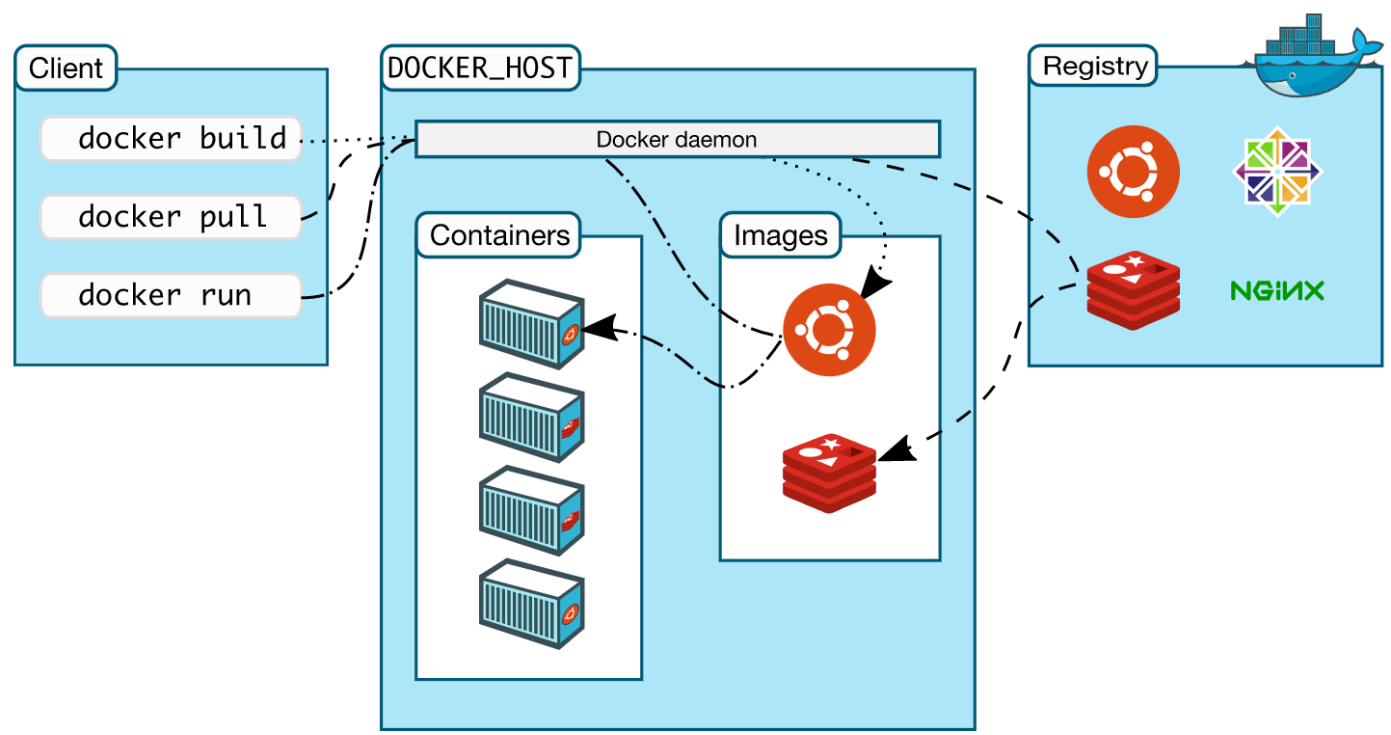
Architecture

(4) Docker objects —— Containers

A container is a runnable instance of an image.

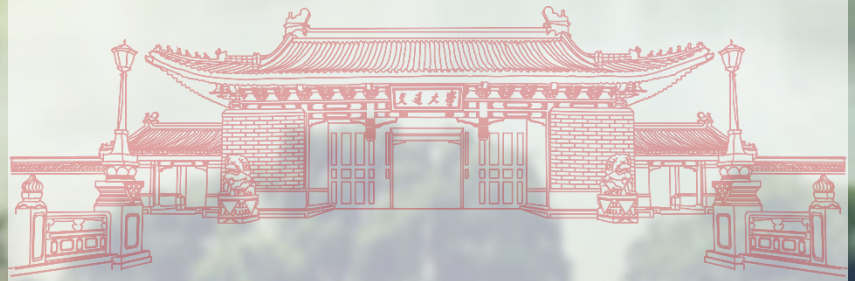
A container is defined by its image as well as any configuration options you provide to it when you create or start it. By default, a container is relatively well isolated from other containers and its host machine.

You can create, start, stop, move, or delete a container using the Docker API or CLI. You can connect a container to one or more networks, attach storage to it, or even create a new image based on its current state. When a container is removed, any changes to its state that are not stored in persistent storage disappear.



02

Use of Docker



Installation

Installation of Docker for Ubuntu

1.Automatic installation using official installation script

>curl -fsSL https://get.docker.com | sudo sh -s

```
chen@chen-virtual-machine: $ curl -fsSL https://get.docker.com | sudo sh -s
# Executing docker install script, commit: 93d2499759296ac1f9c510605fef85052a2c32be
+ sh -c apt-get update -qq >/dev/null
+ sh -c DEBIAN_FRONTEND=noninteractive apt-get install -y -qq apt-transport-https ca-certific
ates curl >/dev/null
+ sh -c curl -fsSL "https://download.docker.com/linux/ubuntu/gpg" | gpg --dearmor --yes -o /u
sr/share/keyrings/docker-archive-keyring.gpg
+ sh -c echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https
://download.docker.com/linux/ubuntu focal stable" > /etc/apt/sources.list.d/docker.list
+ sh -c apt-get update -qq >/dev/null
+ sh -c DEBIAN_FRONTEND=noninteractive apt-get install -y -qq --no-install-recommends  docker
-ce-cli docker-scan-plugin docker-ce >/dev/null
+ version_gte 20.10
+ [ -z  ]
+ return 0
+ sh -c DEBIAN_FRONTEND=noninteractive apt-get install -y -qq docker-ce-rootless-extras >/dev
/null
+ sh -c docker version
Client: Docker Engine - Community
 Version:           20.10.14
 API version:       1.41
 Go version:        go1.16.15
 Git commit:        a224086
 Built:             Thu Mar 24 01:48:02 2022
 OS/Arch:           linux/amd64
 Context:           default
 Experimental:      true

Server: Docker Engine - Community
 Engine:
```

Installation

2.hello-world to test whether successfully installed

>sudo docker run hello-world

```
chen@chen-virtual-machine:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:10d7d58d5ebd2a652f4d93fdd86da8f265f5318c6a73cc5b6a9798ff6d2b2e67
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.
    (amd64)
 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/
```

Installation

3.Modify Docker CGroup driver to systemd

```
>sudo usermod -aG docker $USER
>sudo mkdir -p /etc/docker
>sudo tee /etc/docker/daemon.json <<-'EOF'
{
"exec-opts": ["native.cgroupdriver=systemd"],
"log-driver": "json-file",
"log-opts": {
    "max-size": "100m"
},
"storage-driver": "overlay2",
"registry-mirrors": ["https://docker.mirrors.ustc.edu.cn/"]
}
EOF
```

Installation

4.Load the configuration and restart the docker service

>sudo systemctl daemon-reload

>sudo systemctl restart docker

```
chen@chen-virtual-machine:~$ sudo usermod -aG docker $USER
chen@chen-virtual-machine:~$ sudo mkdir -p /etc/docker
chen@chen-virtual-machine:~$ sudo tee /etc/docker/daemon.json <<-'EOF'
>   {
>   "exec-opts": ["native.cgroupdriver=systemd"],
>   "log-driver": "json-file",
>   "log-opts": {
>       "max-size": "100m"
>   },
>   "storage-driver": "overlay2",
>   "registry-mirrors": ["https://docker.mirrors.ustc.edu.cn/"]
>   }
> EOF
  {
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
      "max-size": "100m"
  },
  "storage-driver": "overlay2",
  "registry-mirrors": ["https://docker.mirrors.ustc.edu.cn/"]
  }
chen@chen-virtual-machine:~$ sudo systemctl daemon-reload
chen@chen-virtual-machine:~$ sudo systemctl restart docker
```

Docker client

Directly enter the *docker* command to view all command options of the docker client

>docker

```
chen@k8s-master:~$ docker

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Options:
      --config string      Location of client config files (default "/home/chen/.docker")
  -c, --context string     Name of the context to use to connect to the daemon (overrides DOCKER_HOST env var and
                           default context set with "docker context use")
  -D, --debug              Enable debug mode
  -H, --host list          Daemon socket(s) to connect to
  -l, --log-level string   Set the logging level ("debug"|"info"|"warn"|"error"|"fatal") (default "info")
      --tls                Use TLS; implied by --tlsverify
      --tlscacert string   Trust certs signed only by this CA (default "/home/chen/.docker/ca.pem")
      --tlscert string     Path to TLS certificate file (default "/home/chen/.docker/cert.pem")
      --tlskey string      Path to TLS key file (default "/home/chen/.docker/key.pem")
      --tlsverify          Use TLS and verify the remote
  -v, --version            Print version information and quit

Management Commands:
  app*        Docker App (Docker Inc., v0.9.1-beta3)
  builder     Manage builds
  buildx*     Docker Buildx (Docker Inc., v0.8.1-docker)
  config      Manage Docker configs
  container   Manage containers
  context     Manage contexts
  image       Manage images
  manifest    Manage Docker image manifests and manifest lists
  network     Manage networks
```

Docker command --help

Through the command *docker command --help*, you can have a deeper understanding of the use of the specified docker command

>docker build --help

```
chen@k8s-master:~$ docker build --help

Usage:  docker build [OPTIONS] PATH | URL | -

Build an image from a Dockerfile

Options:
      --add-host list           Add a custom host-to-IP mapping (host:ip)
      --build-arg list          Set build-time variables
      --cache-from strings      Images to consider as cache sources
      --cgroup-parent string    Optional parent cgroup for the container
      --compress                Compress the build context using gzip
      --cpu-period int          Limit the CPU CFS (Completely Fair Scheduler) period
      --cpu-quota int           Limit the CPU CFS (Completely Fair Scheduler) quota
  -c, --cpu-shares int          CPU shares (relative weight)
      --cpuset-cpus string      CPUs in which to allow execution (0-3, 0,1)
      --cpuset-mems string      MEMs in which to allow execution (0-3, 0,1)
      --disable-content-trust   Skip image verification (default true)
  -f, --file string             Name of the Dockerfile (Default is 'PATH/Dockerfile')
      --force-rm                Always remove intermediate containers
      --iidfile string          Write the image ID to the file
      --isolation string        Container isolation technology
      --label list              Set metadata for an image
  -m, --memory bytes            Memory limit
      --memory-swap bytes       Swap limit equal to memory plus swap: '-1' to enable unlimited swap
      --network string          Set the networking mode for the RUN instructions during build (default "default")
      --no-cache                Do not use cache when building the image
      --pull                    Always attempt to pull a newer version of the image
  -q, --quiet                   Suppress the build output and print image ID on success
      --rm                      Remove intermediate containers after a successful build (default true)
```

Docker images

Use command *docker images* to list images on the local host

>docker images

```
chen@k8s-master:~$ docker images
REPOSITORY                                                       TAG       IMAGE ID       CREATED         SIZE
test_docker                                                      latest    f0bc290b4acf   7 days ago      1.02GB
python                                                           3.9.13    9ac24a438a75   3 weeks ago     915MB
registry.cn-hangzhou.aliyuncs.com/google_containers/kube-apiserver          v1.23.5   3fc1d62d6587   4 months ago    135MB
registry.cn-hangzhou.aliyuncs.com/google_containers/kube-proxy              v1.23.5   3c53fa8541f9   4 months ago    112MB
registry.cn-hangzhou.aliyuncs.com/google_containers/kube-controller-manager v1.23.5   b0c9e5e4dbb1   4 months ago    125MB
registry.cn-hangzhou.aliyuncs.com/google_containers/kube-scheduler          v1.23.5   884d49d6d8c9   4 months ago    53.5MB
rancher/mirrored-flannelcni-flannel                              v0.17.0   9247abf08677   4 months ago    59.8MB
rancher/mirrored-flannelcni-flannel-cni-plugin                   v1.0.1    ac40ce625740   5 months ago    8.1MB
minio/minio                                                      latest    e31e0721a96b   6 months ago    406MB
registry.cn-hangzhou.aliyuncs.com/google_containers/etcd                    3.5.1-0   25f8c7f3da61   8 months ago    293MB
registry.cn-hangzhou.aliyuncs.com/google_containers/coredns                 v1.8.6    a4ca41631cc7   9 months ago    46.8MB
registry.cn-hangzhou.aliyuncs.com/google_containers/pause                   3.6       6270bb605e12   10 months ago   683kB
```

The same *repository* can have multiple *tags*, representing different versions of the repository. For example, there are 15.10, 14.04 and other different versions in the *Ubuntu* repository. We use repository:tag to define different images.

Docker search

You can search the images from docker hub website. (https://hub.docker.com/)
You can also use the *docker search* command to search for images.
>docker search ubuntu

```
chen@k8s-master: $ docker search ubuntu
NAME                              DESCRIPTION                                     STARS     OFFICIAL   AUTOMATED
ubuntu                            Ubuntu is a Debian-based Linux operating sys…   14599     [OK]
websphere-liberty                 WebSphere Liberty multi-architecture images …   286       [OK]
ubuntu-upstart                    DEPRECATED, as is Upstart (find other proces…   112       [OK]
neurodebian                       NeuroDebian provides neuroscience research s…   92        [OK]
open-liberty                      Open Liberty multi-architecture images based…   53        [OK]
ubuntu/nginx                      Nginx, a high-performance reverse proxy & we…   52
ubuntu-debootstrap                DEPRECATED; use "ubuntu" instead                46        [OK]
ubuntu/apache2                    Apache, a secure & extensible open-source HT…   36
ubuntu/mysql                      MySQL open source fast, stable, multi-thread…   34
kasmweb/ubuntu-bionic-desktop     Ubuntu productivity desktop for Kasm Workspa…   29
ubuntu/prometheus                 Prometheus is a systems and service monitori…   27
ubuntu/squid                      Squid is a caching proxy for the Web. Long-t…   25
ubuntu/bind9                      BIND 9 is a very flexible, full-featured DNS…   22
ubuntu/postgres                   PostgreSQL is an open source object-relation…   17
ubuntu/redis                      Redis, an open source key-value store. Long-…   10
ubuntu/grafana                    Grafana, a feature rich metrics dashboard & …   6
ubuntu/prometheus-alertmanager    Alertmanager handles client alerts from Prom…   6
ubuntu/kafka                      Apache Kafka, a distributed event streaming …   6
ubuntu/memcached                  Memcached, in-memory keyvalue store for smal…   5
ubuntu/telegraf                   Telegraf collects, processes, aggregates & w…   4
ubuntu/zookeeper                  ZooKeeper maintains configuration informatio…   4
ubuntu/cortex                     Cortex provides storage for Prometheus. Long…   3
ubuntu/cassandra                  Cassandra, an open source NoSQL distributed …   2
bitnami/ubuntu-base-buildpack     Ubuntu base compilation image                   2                    [OK]
ubuntu/loki                       Grafana Loki, a log aggregation system like …   0
```

Docker pull

To use the image of the official version of *ubuntu*, use the command *docker pull* to download the image.

>docker pull ubuntu

```
chen@k8s-master:~$ docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
7b1a6ab2e44d: Pull complete
Digest: sha256:626ffe58f6e7566e00254b638eb7e0f3b11d4da9675088f4781a50ae288f3322
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
```

```
chen@k8s-master:~$ docker images
REPOSITORY                                                        TAG       IMAGE ID       CREATED         SIZE
test_docker                                                       latest    f0bc290b4acf   7 days ago      1.02GB
python                                                            3.9.13    9ac24a438a75   3 weeks ago     915MB
registry.cn-hangzhou.aliyuncs.com/google_containers/kube-apiserver            v1.23.5   3fc1d62d6587   4 months ago    135MB
registry.cn-hangzhou.aliyuncs.com/google_containers/kube-proxy                v1.23.5   3c53fa8541f9   4 months ago    112MB
registry.cn-hangzhou.aliyuncs.com/google_containers/kube-controller-manager   v1.23.5   b0c9e5e4dbb1   4 months ago    125MB
registry.cn-hangzhou.aliyuncs.com/google_containers/kube-scheduler            v1.23.5   884d49d6d8c9   4 months ago    53.5MB
rancher/mirrored-flannelcni-flannel                               v0.17.0   9247abf08677   4 months ago    59.8MB
rancher/mirrored-flannelcni-flannel-cni-plugin                    v1.0.1    ac40ce625740   5 months ago    8.1MB
minio/minio                                                       latest    e31e0721a96b   6 months ago    406MB
registry.cn-hangzhou.aliyuncs.com/google_containers/etcd                      3.5.1-0   25f8c7f3da61   8 months ago    293MB
ubuntu                                                            latest    ba6acccedd29   9 months ago    72.8MB
registry.cn-hangzhou.aliyuncs.com/google_containers/coredns                   v1.8.6    a4ca41631cc7   9 months ago    46.8MB
registry.cn-hangzhou.aliyuncs.com/google_containers/pause                     3.6       6270bb605e12   10 months ago   683kB
```

Docker rmi

To delete an image, use the *docker rmi* command.

>docker rmi ubuntu

>docker rmi ba6acc

```
chen@k8s-master:~$ docker images
REPOSITORY                                                        TAG       IMAGE ID       CREATED         SIZE
test_docker                                                       latest    f0bc290b4acf   7 days ago      1.02GB
python                                                            3.9.13    9ac24a438a75   3 weeks ago     915MB
registry.cn-hangzhou.aliyuncs.com/google_containers/kube-apiserver            v1.23.5   3fc1d62d6587   4 months ago    135MB
registry.cn-hangzhou.aliyuncs.com/google_containers/kube-proxy                v1.23.5   3c53fa8541f9   4 months ago    112MB
registry.cn-hangzhou.aliyuncs.com/google_containers/kube-controller-manager   v1.23.5   b0c9e5e4dbb1   4 months ago    125MB
registry.cn-hangzhou.aliyuncs.com/google_containers/kube-scheduler            v1.23.5   884d49d6d8c9   4 months ago    53.5MB
rancher/mirrored-flannelcni-flannel                               v0.17.0   9247abf08677   4 months ago    59.8MB
rancher/mirrored-flannelcni-flannel-cni-plugin                    v1.0.1    ac40ce625740   5 months ago    8.1MB
minio/minio                                                       latest    e31e0721a96b   6 months ago    406MB
registry.cn-hangzhou.aliyuncs.com/google_containers/etcd                      3.5.1-0   25f8c7f3da61   8 months ago    293MB
ubuntu                                                            latest    ba6acccedd29   9 months ago    72.8MB
registry.cn-hangzhou.aliyuncs.com/google_containers/coredns                   v1.8.6    a4ca41631cc7   9 months ago    46.8MB
registry.cn-hangzhou.aliyuncs.com/google_containers/pause                     3.6       6270bb605e12   10 months ago   683kB
```

```
chen@k8s-master:~$ docker rmi ubuntu
Untagged: ubuntu:latest
Untagged: ubuntu@sha256:626ffe58f6e7566e00254b638eb7e0f3b11d4da9675088f4781a50ae288f3322
Deleted: sha256:ba6acccedd2923aee4c2acc6a23780b14ed4b8a5fa4e14e252a23b846df9b6c1
Deleted: sha256:9f54eef412758095c8079ac465d494a2872e02e90bf1fb5f12a1641c0d1bb78b
```

Docker run

Docker allows you to run an application in the container.
Use the *docker run* command to build a container based on *ubuntu* image and output “Hello world”.
>docker run ubuntu /bin/echo “Hello world”

```
chen@k8s-master:~$ docker run ubuntu /bin/echo "Hello world"
Hello world
```

What happened?

1.Docker Client passes the *docker run* command to the Docker Engine.
2.Docker Engine creates a new container with *Ubuntu* image.
3.Execute command *bin/echo "Hello world"* in the container, and then output the results.
4.The operation ends and the container stops.

Docker run -it

Use two parameters *-i -t* to let docker run the interactive container.

>docker run -it ubuntu /bin/bash

-t : specify a terminal in the new container

-i : allow interactive operation

```
chen@k8s-master:~$ docker run -it ubuntu /bin/bash
root@d1b685e48d74:/#
```

Run the commands *cat /proc/version* in the container to view the version information of the current system.

```
root@d1b685e48d74:/# cat /proc/version
Linux version 5.13.0-52-generic (buildd@lcy02-amd64-067) (gcc (Ubuntu 9.4.0-1ubun
tu1~20.04.1) 9.4.0, GNU ld (GNU Binutils for Ubuntu) 2.34) #59~20.04.1-Ubuntu SMP
 Thu Jun 16 21:21:28 UTC 2022
```

Run the commands *ls* in the container to view the list of files in the current directory.

```
root@d1b685e48d74:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root
  run  sbin  srv  sys  tmp  usr  var
```

Run the *exit* command or use ctrl+d to exit the container.

```
root@d1b685e48d74:/# exit
exit
chen@k8s-master:~$
```

Docker run -d

Use the *-d* parameter to create a container that runs in the background as a process.
>docker run -d --name=ubuntu-test ubuntu /bin/sh -c "while true; do echo hello world; sleep 1; done"

-d : run in the background
--name : set the name of container

```
chen@k8s-master:~$ docker run -d --name=ubuntu-test ubuntu /bin/sh -c "while true; do echo hello wo
rld; sleep 1; done"
f64bd4347d4737f3440660471b6b836d79513d305f01cd3901ca330e8bf50f50
```

f64bd4 : container ID

Docker ps

To confirm that the container is running, you can check it through command *docker ps*

>docker ps

```
chen@k8s-master: $ docker ps
CONTAINER ID   IMAGE              COMMAND                  CREATED         STATUS         PORTS     NAMES
f64bd4347d47   ubuntu             "/bin/sh -c 'while t…"   3 seconds ago   Up 2 seconds             ubuntu-test
```

- CONTAINER ID: container ID.
- IMAGE: image used.
- COMMAND: the command that runs when the container is started.
- CREATED: the creation time of the container.
- STATUS: container status.
- PORTS: port information of the container and the connection type used (tcp\udp).
- NAMES: automatically assigned container name.

Docker logs

Use the *docker logs* command in the host to view the standard output in the container.

>docker logs ubuntu-test

>docker logs f64bd4

```
chen@k8s-master: $ docker logs ubuntu-test
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
```

Docker attach / exec

When the *-d* parameter is used, the container will enter the background after starting. At this time, you can enter the container through the following command:

>docker attach ubuntu-test

>docker attach f21d01

```
chen@k8s-master:~$ docker run -itd --name=ubuntu-test ubuntu /bin/bash
f21d01533a462dad951e8156a43217442f009b62a45760f62e340ef2ba474109
chen@k8s-master:~$ docker attach f21d01
root@f21d01533a46:/# exit
exit
```

stopped

>docker exec -it ubuntu-test /bin/bash

>docker exec -it f21d01 /bin/bash

```
chen@k8s-master:~$ docker exec -it f21d01 /bin/bash
root@f21d01533a46:/# exit
exit
```

>docker exec ubuntu-test echo “Hello world”

>docker exec f21d01 “Hello world”

```
chen@k8s-master:~$ docker exec f21d01 echo "Hello world"
Hello world
```

Docker stop / restart / rm

To stop a container, use command *docker stop*.

>docker stop ubuntu-test

>docker stop f64bd4

```
chen@k8s-master:~$ docker stop ubuntu-test
ubuntu-test
```

To restart a container, use command *docker restart*.

>docker restart ubuntu-test

>docker restart f64bd4

```
chen@k8s-master:~$ docker restart ubuntu-test
ubuntu-test
```

To remove a container, use command *docker rm*.

>docker rm ubuntu-test

>docker rm f64bd4

```
chen@k8s-master:~$ docker rm ubuntu-test
ubuntu-test
```

To remove all the stopped container, use following command

>docker container prune

Docker container connection

Network applications can be run in the container. To allow external access to these applications, you can specify the port mapping through the -P or -p parameter.

>docker run -d -P training/webapp python app.py

>docker run -d -p 5000:5000 training/webapp python app.py

>docker run -d -p 127.0.0.1:5001:5000 training/webapp python app.py

>docker run -d -p 127.0.0.1:5000:5000/udp training/webapp python app.py

-P : randomly map the network port used inside the container to the host

-p : map the network port used inside the container to the specified host port

```
chen@k8s-master:~$ docker ps
CONTAINER ID   IMAGE                                                COMMAND                  CREATED
     STATUS          PORTS                                        NAMES
505c841678d8   training/webapp                                      "python app.py"          3 seconds ag
o    Up 2 seconds    5000/tcp, 127.0.0.1:5000->5000/udp           strange_montalcini
e91e42c29ea5   training/webapp                                      "python app.py"          7 minutes ag
o    Up 7 minutes    127.0.0.1:5001->5000/tcp                     upbeat_colden
c1bef439a2c1   training/webapp                                      "python app.py"          7 minutes ag
o    Up 7 minutes    0.0.0.0:5000->5000/tcp, :::5000->5000/tcp    romantic_wescoff
e5d9624d2624   training/webapp                                      "python app.py"          29 minutes a
go   Up 29 minutes   0.0.0.0:49154->5000/tcp, :::49154->5000/tcp  loving_beaver
```

The *docker port* command allows us to quickly view the mapping of ports.

>docker port e91e42 5000

```
chen@k8s-master:~$ docker port e91e42 5000
127.0.0.1:5001
```

Docker commit

When the image we downloaded from the docker image repository cannot meet our needs, we can change the image in the following two ways:

1.Update the image from the container that has been created and commit the image

>docker run -it ubuntu /bin/bash
>apt-get update
>exit
>docker commit -m="has update"
  -a="chen" b7e719 chen/ubuntu:v2

-m : commit info
-a : author
b7e719 : container ID
chen/ubuntu:v2 : target image name

```
chen@k8s-master:~$ docker run -it ubuntu /bin/bash
root@b7e7197baa54:/# apt-get update
Get:1 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:2 http://archive.ubuntu.com/ubuntu focal InRelease [265 kB]
Get:3 http://security.ubuntu.com/ubuntu focal-security/universe amd64 Packages [883 kB]
Get:4 http://archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:5 http://archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Get:6 http://archive.ubuntu.com/ubuntu focal/multiverse amd64 Packages [177 kB]
Get:7 http://archive.ubuntu.com/ubuntu focal/main amd64 Packages [1275 kB]
Get:8 http://security.ubuntu.com/ubuntu focal-security/restricted amd64 Packages [1398 kB]
Get:9 http://archive.ubuntu.com/ubuntu focal/universe amd64 Packages [11.3 MB]
Get:10 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [2027 kB]
Get:11 http://security.ubuntu.com/ubuntu focal-security/multiverse amd64 Packages [27.5 kB]
Get:12 http://archive.ubuntu.com/ubuntu focal/restricted amd64 Packages [33.4 kB]
Get:13 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [2475 kB]
Get:14 http://archive.ubuntu.com/ubuntu focal-updates/multiverse amd64 Packages [30.2 kB]
Get:15 http://archive.ubuntu.com/ubuntu focal-updates/universe amd64 Packages [1161 kB]
Get:16 http://archive.ubuntu.com/ubuntu focal-updates/restricted amd64 Packages [1511 kB]
Get:17 http://archive.ubuntu.com/ubuntu focal-backports/universe amd64 Packages [27.1 kB]
Get:18 http://archive.ubuntu.com/ubuntu focal-backports/main amd64 Packages [54.2 kB]
Fetched 23.0 MB in 8s (2944 kB/s)
Reading package lists... Done
root@b7e7197baa54:/# exit
exit
chen@k8s-master:~$ docker commit -m="has update" -a="chen" b7e719 chen/ubuntu:v2
sha256:97fa15c151f1b82812d2c8a8a2a820a56f3bff8242bb5ef4436bfe266ad17beb
chen@k8s-master:~$ docker images
REPOSITORY                                                        TAG       IMAGE
chen/ubuntu                                                       v2        97fa1
```

Dockerfile

When the image we downloaded from the docker image repository cannot meet our needs, we can change the image in the following two ways:

2.Use the *dockerfile* to create a new image

Dockerfile is a text file used to build images. The text contains commands and instructions required for building images.

Every time the dockerfile instruction is executed, a new layer will be created on the docker.

In an empty directory, create a new file named *Dockerfile*, and add the following contents to the file：

FROM nginx

RUN echo '这是一个本地构建的nginx镜像' > /usr/share/nginx/html/index.html

```
chen@k8s-master:~$ mkdir Dockerfile
chen@k8s-master:~$ cd Dockerfile
chen@k8s-master:~/Dockerfile$ vi Dockerfile
```

```
FROM nginx
RUN echo '这是一个本地构建的nginx镜像' > /usr/share/nginx/html/index.html
```

Docker build

Execute the command *docker build* under the directory of *dockerfile.*

>docker build -t nginx:v3 .

-t : reporitory:tag

. : context path of the execution

```
chen@k8s-master:~/Dockerfile$ docker build -t nginx:v3 .
Sending build context to Docker daemon  2.048kB
Step 1/2 : FROM nginx
latest: Pulling from library/nginx
latest: Pulling from library/nginx
461246efe0a7: Pull complete
a96aaf9a9ec3: Pull complete
650d8b758441: Pull complete
b138da793ac8: Pull complete
bb1705539683: Pull complete
b9ed43dcc388: Pull complete
Digest: sha256:db345982a2f2a4257c6f699a499feb1d79451a1305e8022f16456ddc3ad6b94c
Status: Downloaded newer image for nginx:latest
 ---> 41b0e86104ba
Step 2/2 : RUN echo '这是一个本地构建的nginx镜像' > /usr/share/nginx/html/index.html
 ---> Running in 0368d3495856
Removing intermediate container 0368d3495856
 ---> aca2bbf226a3
Successfully built aca2bbf226a3
Successfully tagged nginx:v3
chen@k8s-master:~/Dockerfile$ docker images
REPOSITORY                                                        TAG        IMAGE ID       CREATED
SIZE
nginx                                                             v3         aca2bbf226a3   About a minute ago
142MB
```

Dockerfile instructions

FROM

Customized images are based on FROM images. Nginx here is the basic image required for customization.

FROM <image>

RUN

Execute the following command-line commands.

RUN <command-line command>

<command-line command > is equivalent to the shell command operated on the terminal.

RUN ["<executable>", "<param1>", "<param2>"]

For example: *RUN ["./test.php", "dev", "offline"]* is equivalent to *RUN ./test.php dev offline*

WORKDIR

Specify the working directory. The working directory specified with workdir will exist in every layer of the image.(the working directory specified by WORKDIR must be created in advance)

WORKDIR <working directory path>

Dockerfile instructions

COPY

Copy files or directories from the context directory to the specified path in the container.

COPY [--chown=<user>:<group>] <source path1>... <destination path>

COPY [--chown=<user>:<group>] ["<source path1>",... "<destination path>"]

CMD

Similar to the RUN instruction, used to run programs.

CMD <command-line command>

CMD ["<executable>",">param1>","<param2>",...]

CMD ["<param1>","<param2>",...]

Provide default parameters for the program specified by the ENTRYPOINT instruction.

Diff:

1.run time: CMD docker run; RUN docker build.

2.The program specified by CMD instruction can be overwritten by the program specified in *docker run* command.

3.If there are multiple CMD instructions in dockerfile, only the last one works.

Dockerfile instructions

ENTRYPOINT

Similar to CMD instruction, but it will not be overwritten by the instruction specified by the command line parameters of *docker run*, and these command line parameters will be sent to the program specified by the ENTRYPOINT instruction as parameters.

If there are multiple ENTRYPOINT instructions in dockerfile, only the last one works.

ENTRYPOINT ["<executeable>","<param1>","<param2>",...]

Assume that the nginx:test image has been built through dockerfile:

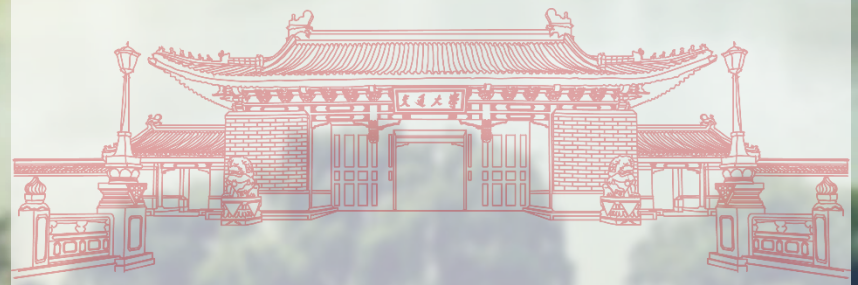
```
FROM nginx
ENTRYPOINT ["nginx", "-c"]
CMD ["/etc/nginx/nginx.conf"]
```

```
>docker run  nginx:test
nginx -c /etc/nginx/nginx.conf
```

```
>docker run  nginx:test -c /etc/nginx/new.conf
nginx -c /etc/nginx/new.conf
```

03

Introduction to Kubernetes



Overview

What is Kubernetes?

Kubernetes is a portable, extensible, open source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. Kubernetes provides some generally applicable features common to PaaS offerings, such as deployment, scaling, load balancing, and lets users integrate their logging, monitoring, and alerting solutions. But Kubernetes operates at the container level rather than at the hardware level.

Why is Kubernetes?

Containers are a good way to bundle and run your applications. In a production environment, you need to manage the containers that run the applications and ensure that there is no downtime. For example, if a container goes down, another container needs to start.

Kubernetes provides you with a framework to run distributed systems resiliently. It takes care of scaling and failover for your application, provides deployment patterns, and more.

Overview

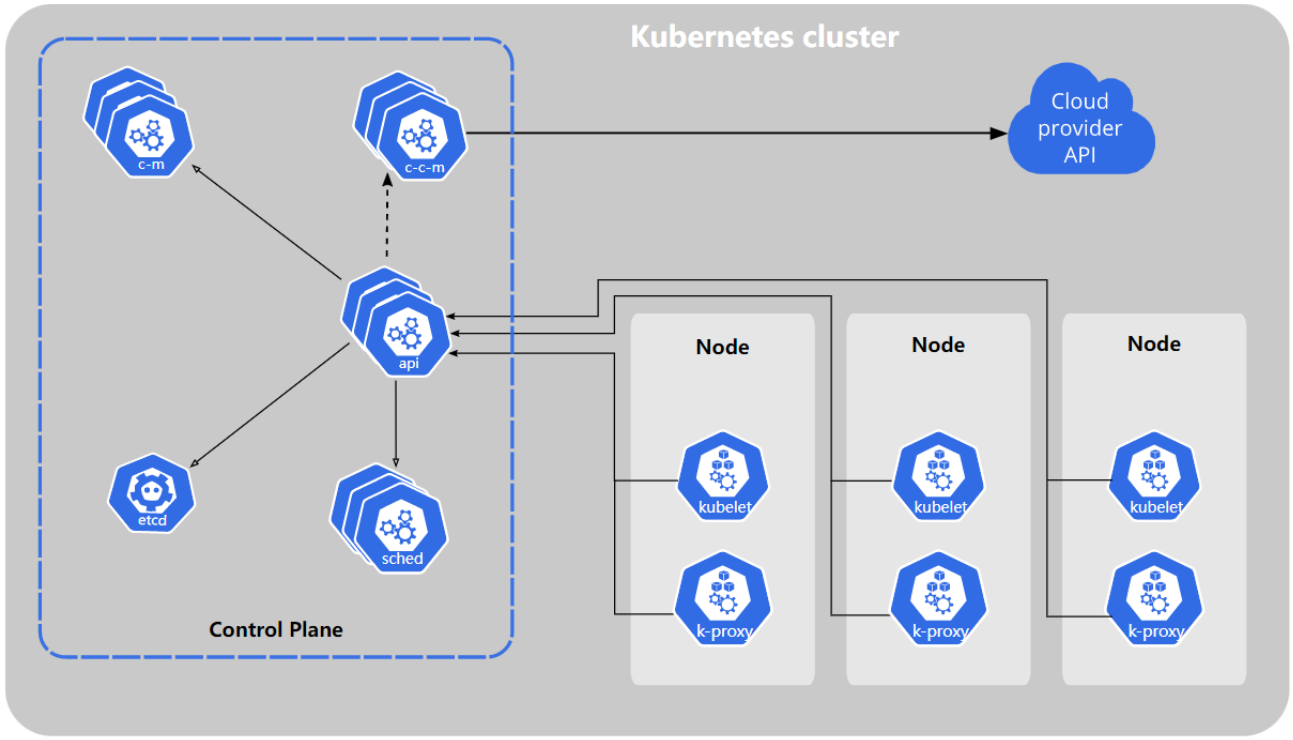
Kubernetes provide:

- Service discovery and load balancing
- Storage orchestration
- Automated rollouts and rollbacks
- Automatic bin packing
- Self-healing
- Secret and configuration management

Kubernetes cluster

A Kubernetes cluster consists of a set of worker machines, called *nodes*, that run containerized applications. Every cluster has at least one worker node.

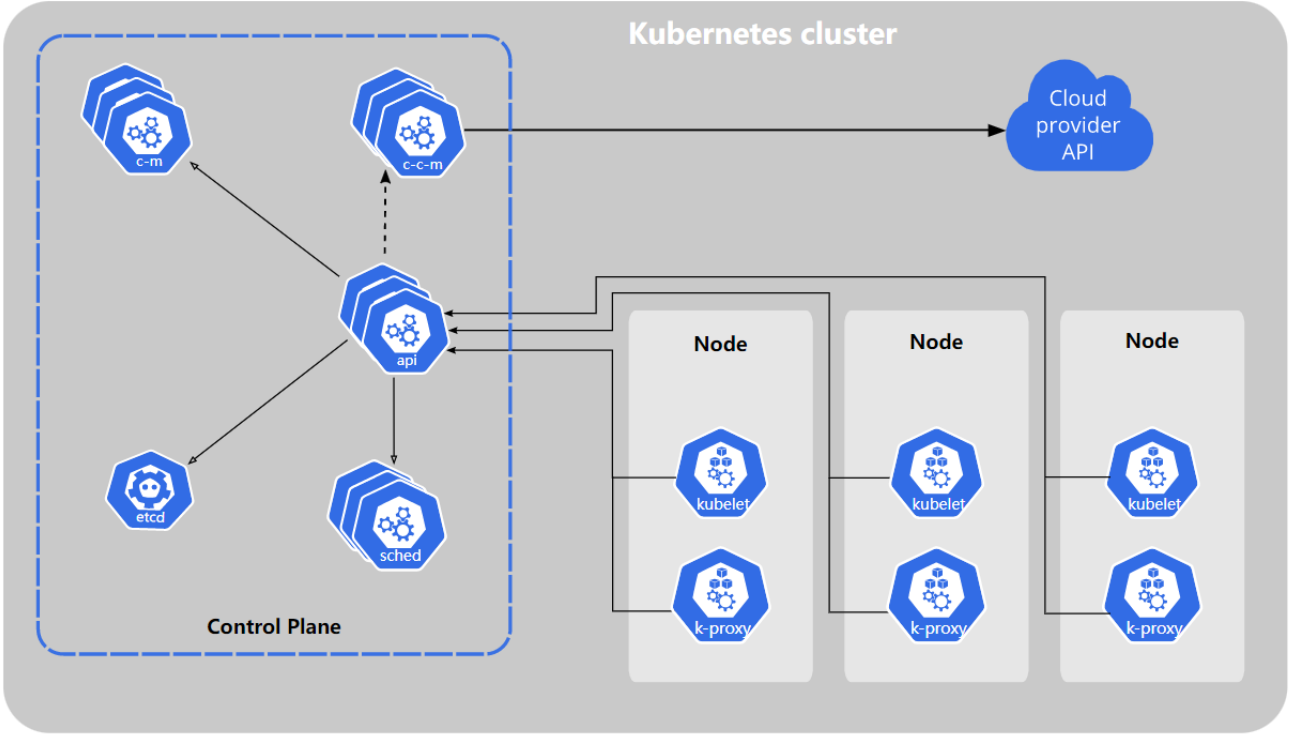
- The worker node(s) host the Pods that are the components of the application workload.
- The control plane manages the worker nodes and the Pods in the cluster.



In production environments, the control plane usually runs across multiple computers and a cluster usually runs multiple nodes, providing fault-tolerance and high availability.

Control Plane Components

The control plane's components make global decisions about the cluster (for example, scheduling), as well as detecting and responding to cluster events (for example, starting up a new pod when a deployment's replicas field is unsatisfied)



1.kube-apiserver

The API server is a component of the Kubernetes control plane that exposes the Kubernetes API. The API server is the front end for the Kubernetes control plane.

2.etcd

Consistent and highly-available key value store used as Kubernetes' backing store for all cluster data.

Control Plane Components

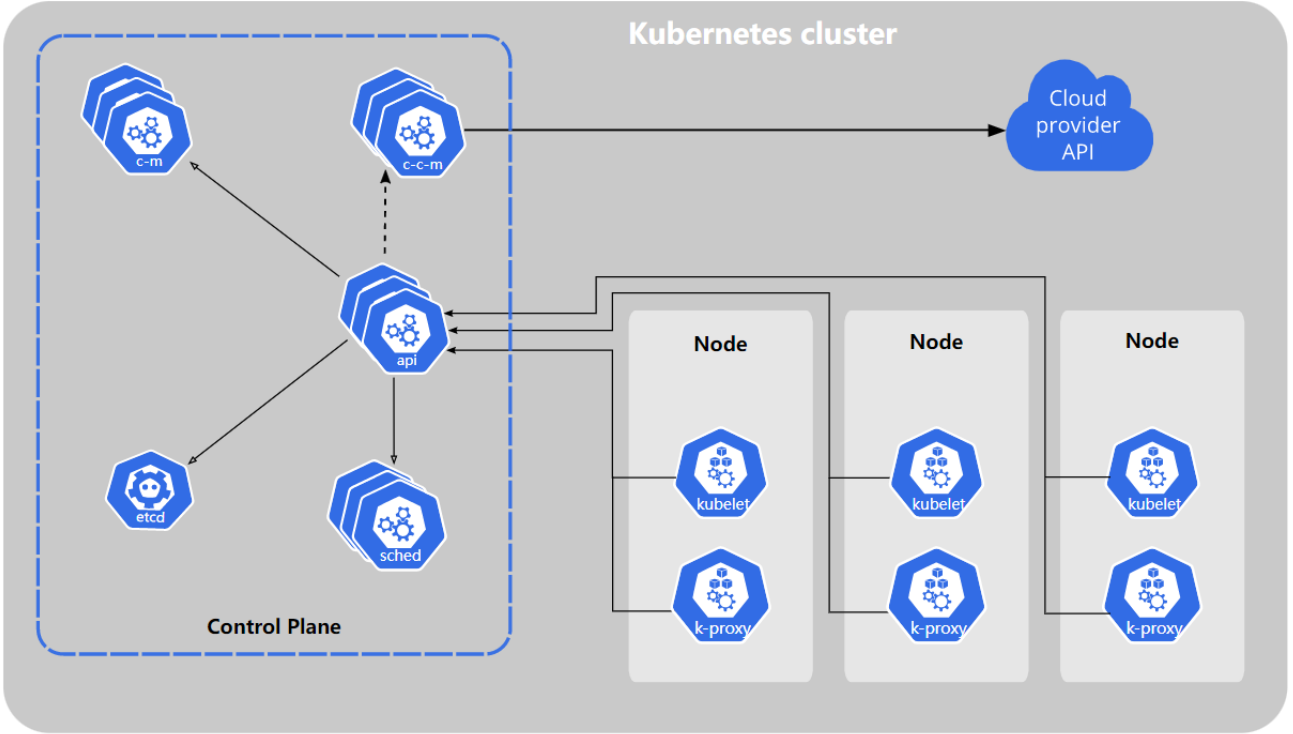
3.kube-scheduler

Control plane component that watches for newly created Pods with no assigned node, and selects a node for them to run on.

4.kube-controller-manager

Control plane component that runs controller processes.

- Node controller
- Job controller
- Endpoints controller
- Service Account & Token controllers

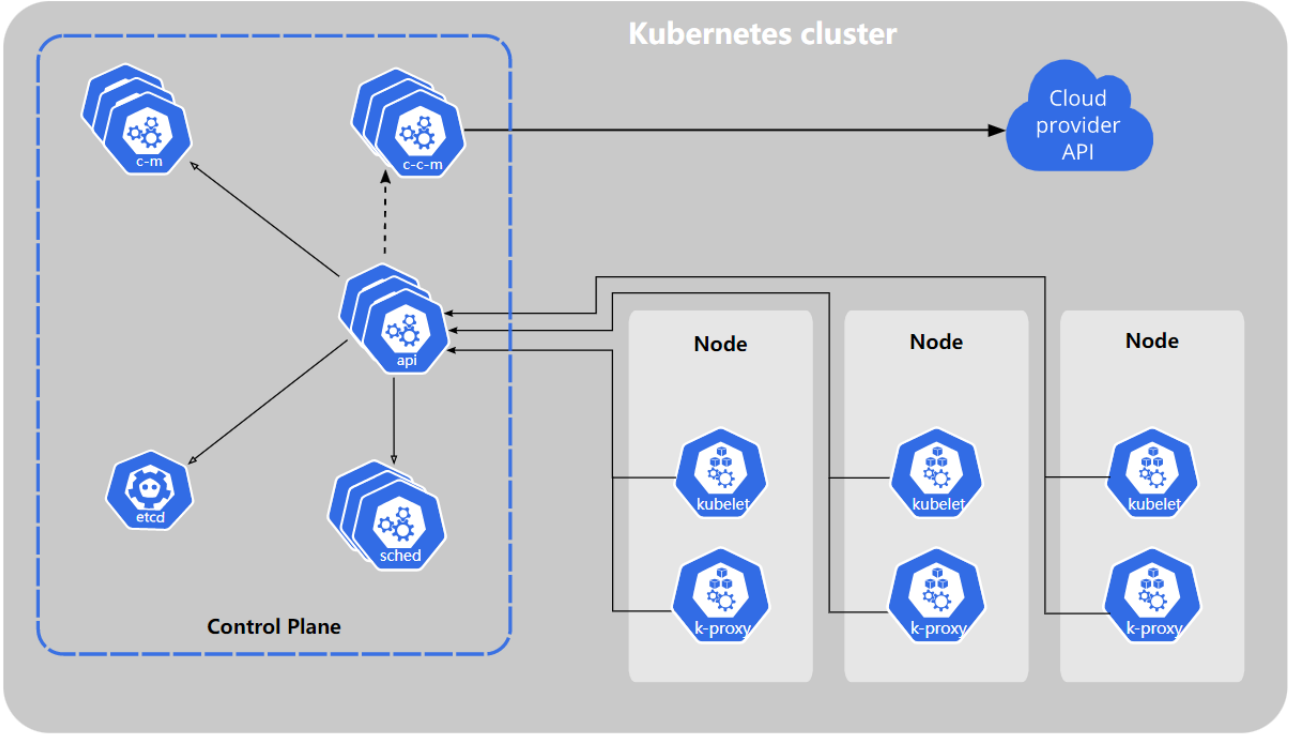


Control Plane Components

5.cloud-controller-manager

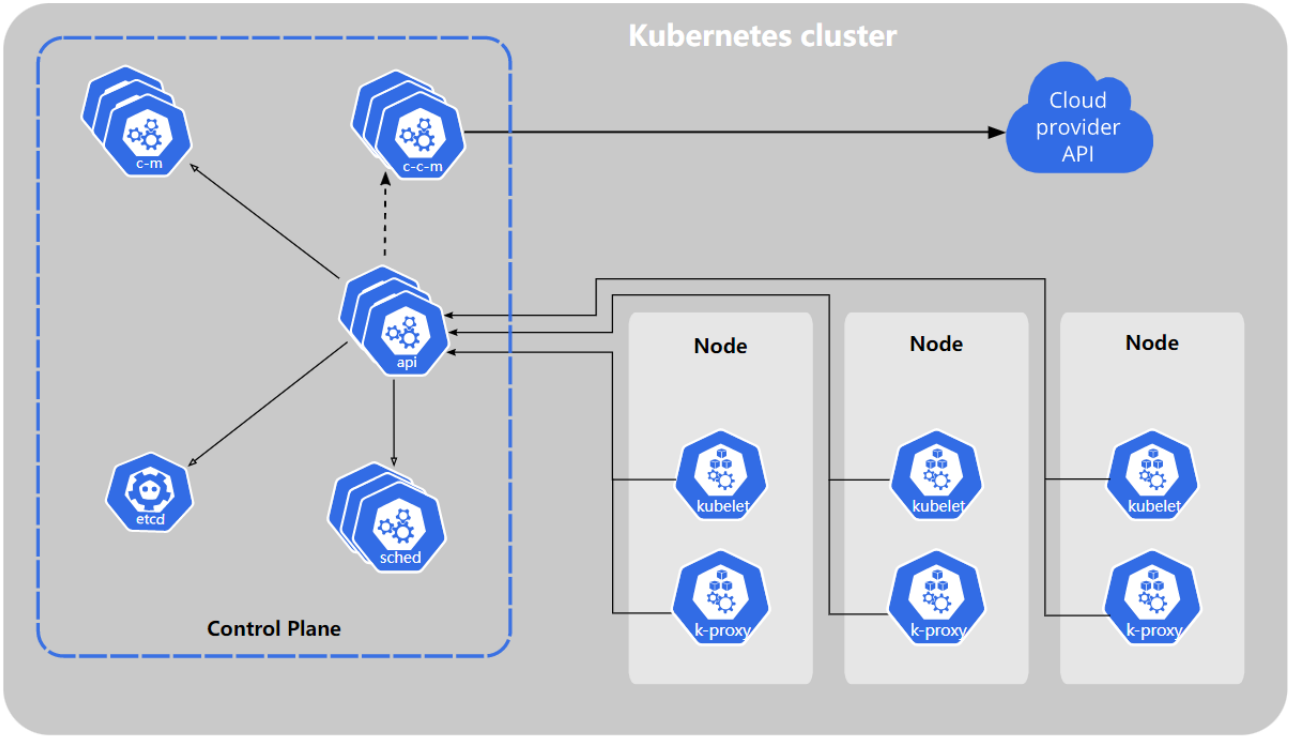
A Kubernetes control plane component that embeds cloud-specific control logic. The cloud controller manager lets you link your cluster into your cloud provider's API, and separates out the components that interact with that cloud platform from components that only interact with your cluster.

- Node controller
- Route controller
- Service controller



Node Components

The node components run on every node, maintaining running pods and providing the Kubernetes runtime environment.



1.kubelet

An agent that runs on each node in the cluster. It makes sure that containers are running in a Pod.

2.kube-proxy

kube-proxy is a network proxy that runs on each node in your cluster, implementing part of the Kubernetes Service concept.

3.Container runtime

The container runtime is the software that is responsible for running containers. Kubernetes supports container runtimes such as docker, containerd, CRI-O, and any other implementation of the Kubernetes CRI (Container Runtime Interface)

Kubernetes Objects

Kubernetes objects are persistent entities in the Kubernetes system. Kubernetes uses these entities to represent the state of your cluster.

Specifically, they can describe:
1.What containerized applications are running (and on which nodes)
2.The resources available to those applications
3.The policies around how those applications behave, such as restart policies, upgrades, and fault-tolerance

A Kubernetes object is a "record of intent"--once you create the object, the Kubernetes system will constantly work to ensure that object exists. By creating an object, you're effectively telling the Kubernetes system what you want your cluster's workload to look like; this is your cluster's **desired state**.

Object Spec and Status

Almost every Kubernetes object includes two nested object fields that govern the object's configuration: the **object spec** and the **object status**.

For objects that have a **spec**, you have to set this when you create the object, providing a description of the characteristics you want the resource to have: its desired state.

The **status** describes the current state of the object, supplied and updated by the Kubernetes system and its components. The Kubernetes control plane continually and actively manages every object's actual state to match the desired state you supplied.

When you create an object in Kubernetes, you must provide the object spec that describes its desired state, as well as some basic information about the object (such as a name). When you use the Kubernetes API to create the object (either directly or via *kubectl*), that API request must include that information as JSON in the request body. **Most often, you provide the information to *kubectl* in a .yaml file.** *kubectl* converts the information to JSON when making the API request.

.yaml file

Here's an example *.yaml* file that shows the required fields and object spec for a Kubernetes Deployment: (In Kubernetes, a Deployment is an object that can represent an application running on your cluster.)

application/deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2 # tells deployment to run 2 pods matching the template
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

Required Fields:

- **apiVersion** - which version of the Kubernetes API you're using to create this object
- **kind** - what kind of object you want to create
- **metadata** - wata that helps uniquely identify the object, including a name string, UID, and optional namespace
- **spec** - what state you desire for the object

Pod

Pods are the smallest deployable units of computing that you can create and manage in Kubernetes.

A Pod is a group of one or more containers, with shared storage and network resources, and a specification for how to run the containers. A Pod's contents are always co-located and co-scheduled, and run in a shared context. A Pod models an application-specific "logical host": it contains one or more application containers which are relatively tightly coupled.

The shared context of a Pod is a set of Linux namespaces, cgroups, and potentially other facets of isolation - the same things that isolate a Docker container. Within a Pod's context, the individual applications may have further sub-isolations applied.

Pod

The following is an example of a Pod which consists of a container running the image nginx:1.14.2:

pods/simple-pod.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
```

Pods in a Kubernetes cluster are used in two main ways:

- Pods that run a single container.
- Pods that run multiple containers that need to work together.

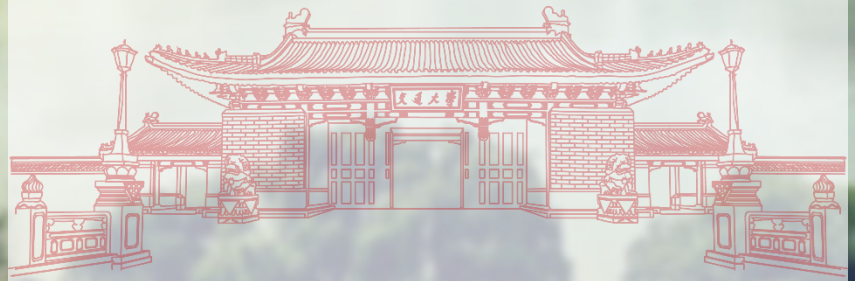
If you want to scale your application horizontally (to provide more overall resources by running more instances), you should use multiple Pods, one for each instance. In Kubernetes, this is typically referred to as replication.

Workload Resources

- **Deployments**:represent an application running on your cluster.
- **ReplicaSet**:to maintain a stable set of replica Pods running at any given time.
- **StatefulSet**:to manage stateful applications.
- **DaemonSet**:to ensures that all (or some) Nodes run a copy of a Pod.
- **Jobs**:creates one or more Pods and will continue to retry execution of the Pods until a specified number of them successfully terminate.

04

Use of Kubernetes



Installation

1.Add and trust apt certificate

>curl https://mirrors.aliyun.com/kubernetes/apt/doc/apt-key.gpg | sudo apt-key add -

Add source address

>sudo add-apt-repository "deb https://mirrors.aliyun.com/kubernetes/apt/ kubernetes-xenial main"

```
chen@chen-virtual-machine:~$ curl https://mirrors.aliyun.com/kubernetes/apt/doc/apt-key.gpg | sudo apt-ke
y add -
[sudo] password for chen:   % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  2537  100  2537    0     0  16159      0 --:--:-- --:--:-- --:--:-- 16159

OK
chen@chen-virtual-machine:~$ sudo add-apt-repository "deb https://mirrors.aliyun.com/kubernetes/apt/ kube
rnetes-xenial main"
Get:1 https://mirrors.aliyun.com/kubernetes/apt kubernetes-xenial InRelease [9,383 B]
Hit:2 https://mirrors.ustc.edu.cn/docker-ce/linux/ubuntu focal InRelease
Hit:3 https://download.docker.com/linux/ubuntu focal InRelease
Ign:4 https://mirrors.aliyun.com/kubernetes/apt kubernetes-xenial/main amd64 Packages
Hit:5 http://cn.archive.ubuntu.com/ubuntu focal InRelease
Get:6 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:4 https://mirrors.aliyun.com/kubernetes/apt kubernetes-xenial/main amd64 Packages [54.7 kB]
Get:7 http://cn.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
```

Installation

2.Update and install

>sudo apt update && sudo apt install -y kubelet kubeadm kubectl

```
chen@chen-virtual-machine:~$ sudo apt update && sudo apt install -y kubelet kubeadm kubectl
Hit:1 https://mirrors.aliyun.com/kubernetes/apt kubernetes-xenial InRelease
Hit:2 https://mirrors.ustc.edu.cn/docker-ce/linux/ubuntu focal InRelease
Hit:3 https://download.docker.com/linux/ubuntu focal InRelease
Hit:4 http://security.ubuntu.com/ubuntu focal-security InRelease
Hit:5 http://cn.archive.ubuntu.com/ubuntu focal InRelease
Hit:6 http://cn.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:7 http://cn.archive.ubuntu.com/ubuntu focal-backports InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
132 packages can be upgraded. Run 'apt list --upgradable' to see them.
W: Target Packages (stable/binary-amd64/Packages) is configured multiple times in /etc/apt/sources.
```

3.Add completion

>source <(kubectl completion bash)

>source <(kubeadm completion bash)

```
chen@chen-virtual-machine:~$ source <(kubectl completion bash)
chen@chen-virtual-machine:~$ source <(kubeadm completion bash)
chen@chen-virtual-machine:~$
```

Build k8s cluster

1.Modify host configuration
Execute the following two commands on Master and Worker respectively to edit the corresponding file

>sudo vim /etc/hostname
>sudo vim /etc/hosts

master
> cat /etc/hosts
192.168.124.129 k8s-master
192.168.124.131 k8s-worker
> cat /etc/hostname
k8s-master
worker
> cat /etc/hosts
192.168.124.129 k8s-master
192.168.124.131 k8s-worker
> cat /etc/hostname
k8s-worker

```
chen@chen-virtual-machine:~$ sudo vim /etc/hostname
chen@chen-virtual-machine:~$ sudo vim /etc/hosts
chen@chen-virtual-machine:~$ chen@chen-virtual-machine:~$
chen@chen-virtual-machine:~$ cat /etc/hostname
k8s-master
chen@chen-virtual-machine:~$ cat /etc/hosts
192.168.159.129 k8s-master
192.168.159.130 k8s-worker
```

Build k8s cluster

2.Edit the */etc/fstab* file to prohibit the exchange of partitions

>vim /etc/fstab

/swapfile none swap sw 0 0

Perform this operation on the Master and Worker respectively, and restart the machine.

```
chen@chen-virtual-machine:~$ sudo vim /etc/fstab
```

3.Execute the following commands on the Master

>sudo kubeadm init --apiserver-advertise-address <Master_IP> \

--pod-network-cidr=10.244.0.0/16 \

--image-repository registry.cn-hangzhou.aliyuncs.com/google_containers

<Master_IP> is the ip address of the Master

```
chen@k8s-master:~$ sudo kubeadm init --apiserver-advertise-address 192.168.159.129 \
> --pod-network-cidr=10.244.0.0/16  \
> --image-repository registry.cn-hangzhou.aliyuncs.com/google_containers
```

Build k8s cluster

4.The Master initialization is successful.
The last part of the output information, corresponding to the command of Worker nodes to join the cluster,needs to be saved.

```
Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 192.168.159.129:6443 --token jb7jwz.bec6h580w1p9q9wa \
        --discovery-token-ca-cert-hash sha256:869f7cb8c60c2aa153eb11c93a97b64adac739b6bc19c354636460eb87211e0a
```

Build k8s cluster

5.Follow the prompts in the output to execute the following commands:

>sudo mkdir -p $HOME/.kube

>sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

>sudo chown $(id -u):$(id -g) $HOME/.kube/config

```
chen@k8s-master:~$ mkdir -p $HOME/.kube
chen@k8s-master:~$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
[sudo] password for chen:
chen@k8s-master:~$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Configure the environment, then the Master can execute the *kubectl* command

6. Try to execute the following command:

>kubectl get node

```
chen@k8s-master:~$ kubectl get node
NAME         STATUS     ROLES                  AGE   VERSION
k8s-master   NotReady   control-plane,master   83m   v1.23.5
```

The status shows *NotReady* because the network has not been configured

Build k8s cluster

7.Execute the following command to install the flannel network plug-in:

>kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

```
chen@k8s-master:~$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/
kube-flannel.yml
Warning: policy/v1beta1 PodSecurityPolicy is deprecated in v1.21+, unavailable in v1.25+
podsecuritypolicy.policy/psp.flannel.unprivileged created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
```

8.Execute the previously saved command on the Worker to join the cluster

>kubeadm join 192.168.159.129:6443 --token jb7jwz.bec6h580w1p9q9wa \
--discovery-token-ca-cert-hash sha256:869f7cb8c60c2aa153eb11c93a97b64adac739b6bc19c354636460eb87211e0a

```
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...
[kubelet-check] Initial timeout of 40s passed.


This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

Build k8s cluster

9.Check the nodes on the Master

>kubectl get nodes

```
chen@k8s-master:~$ kubectl get nodes
NAME         STATUS   ROLES                  AGE    VERSION
k8s-master   Ready    control-plane,master   112m   v1.23.5
k8s-worker   Ready    <none>                 2m5s   v1.23.5
```

The Worker node has joined the cluster, and the node status has changed to *Ready*

10.Check the correctness of pod

>kubectl get pods -A

```
chen@k8s-master:~$ kubectl get pods -A
NAMESPACE     NAME                                 READY   STATUS    RESTARTS        AGE
kube-system   coredns-65c54cc984-hxlps             1/1     Running   2 (10m ago)     113m
kube-system   coredns-65c54cc984-kcwpc             1/1     Running   2 (10m ago)     113m
kube-system   etcd-k8s-master                      1/1     Running   0               114m
kube-system   kube-apiserver-k8s-master            1/1     Running   1               114m
kube-system   kube-controller-manager-k8s-master   1/1     Running   4 (4m34s ago)   114m
kube-system   kube-flannel-ds-6zfgb                1/1     Running   0               17m
kube-system   kube-flannel-ds-r72cm                1/1     Running   0               3m27s
kube-system   kube-proxy-ckbzv                     1/1     Running   0               113m
kube-system   kube-proxy-fgzg2                     1/1     Running   0               3m27s
kube-system   kube-scheduler-k8s-master            1/1     Running   3 (5m11s ago)   114m
```

Create a Deployment

The following is an example of a Deployment.
It creates a ReplicaSet to bring up three nginx Pods:

controllers/nginx-deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

Create a Deployment

Before starting, make sure that the kubernetes cluster of is up and running.

>kubectl get nodes

```
chen@k8s-master:~$ kubectl get nodes
NAME          STATUS   ROLES                  AGE     VERSION
k8s-master    Ready    control-plane,master   3d21h   v1.23.5
k8s-worker    Ready    <none>                 3d19h   v1.23.5
```

Edit *nginx_deployment.yaml* file

>vim nginx_deployment.yaml

```
chen@k8s-master:~$ cd nginx
chen@k8s-master:~/nginx$ ls
nginx_deployment.yaml
chen@k8s-master:~/nginx$ vim nginx_deployment.yaml
```

Create a deployment by running the following command:

>kubectl apply -f nginx_deployment.yaml

```
chen@k8s-master:~/nginx$ kubectl apply -f nginx_deployment.yaml
deployment.apps/nginx-deployment created
```

>kubectl apply -f https://k8s.io/examples/controllers/nginx-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

Create a Deployment

Check whether the deployment has been created

>kubectl get deployment

```
chen@k8s-master:~/nginx$ kubectl get deployment
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   0/2     2            0           4s
```

- NAME:name of the deployment in the cluster
- READY:the number of "copies" available for the application. The displayed mode is "ready number / expected number"
- UP-TO-DATE:the number of copies that have been updated in order to reach the desired state
- AVAILABLE: the number of copies of the application available to the user
- AGE: the time the application was running

To see the Deployment rollout status

>kubectl rollout status deployment/nginx-deployment

```
chen@k8s-master:~/nginx$ kubectl rollout status deployment/nginx-deployment
Waiting for deployment "nginx-deployment" rollout to finish: 0 of 2 updated replicas are available...
```

Create a Deployment

Get details of the Deployment

>kubectl describe deployment nginx-deployment

```
chen@k8s-master: /nginx$ kubectl describe deployment nginx-deployment
Name:                   nginx-deployment
Namespace:              default
CreationTimestamp:      Fri, 22 Apr 2022 09:51:27 +0800
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=nginx
Replicas:               2 desired | 2 updated | 2 total | 2 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=nginx
  Containers:
   nginx:
    Image:        nginx:1.14.2
    Port:         80/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   nginx-deployment-9456bbbf9 (2/2 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  17s   deployment-controller  Scaled up replica set nginx-deployment-9456bbbf9 to 2
```

Create a Deployment

To see the ReplicaSet (rs) created by the Deployment

>kubectl get rs.

```
chen@k8s-master: /nginx$ kubectl get rs
NAME                          DESIRED   CURRENT   READY   AGE
nginx-deployment-9456bbbf9    2         2         0       2s
```

- NAME: the names of the ReplicaSets in the namespace
- DESIRED: the desired number of replicas of the application, which you define when you create the Deployment. This is the desired state
- CURRENT: how many replicas are currently running
- READY: how many replicas of the application are available to your users
- AGE: the amount of time that the application has been running

To check the Pods in the Deployment

>kubectl get pod -l app=nginx

```
chen@k8s-master: /nginx$ kubectl get pod -l app=nginx
NAME                          READY   STATUS    RESTARTS   AGE
nginx-deployment-9456bbbf9-2zrxr   1/1     Running   0          50s
nginx-deployment-9456bbbf9-n5mbk   1/1     Running   0          50s
```

Create a Deployment

Get details of the Pod

>kubectl describe pod <Pod-Name>

```
chen@k8s-master:~/nginx$ kubectl describe pod nginx-deployment-9456bbbf9-2zrxr
Name:         nginx-deployment-9456bbbf9-2zrxr
Namespace:    default
Priority:     0
Node:         k8s-worker/192.168.159.130
Start Time:   Fri, 22 Apr 2022 09:51:27 +0800
Labels:       app=nginx
              pod-template-hash=9456bbbf9
Annotations:  <none>
Status:       Running
IP:           10.244.1.36
IPs:
  IP:           10.244.1.36
Controlled By:  ReplicaSet/nginx-deployment-9456bbbf9
Containers:
  nginx:
    Container ID:   docker://f459c66912274db95908c1ac8cb7ebb904ab292575f38a12a9da951e593099bc
    Image:          nginx:1.14.2
    Image ID:       docker-pullable://nginx@sha256:f7988fb6c02e0ce69257d9bd9cf37ae20a60f1df7563c3a2a6abe24160306b8d
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Fri, 22 Apr 2022 09:51:33 +0800
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-qmrd5 (ro)
```

Create a Deployment

Get details of the Pod

>kubectl describe pod <Pod-Name>

```
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  kube-api-access-qmrd5:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  66s   default-scheduler  Successfully assigned default/nginx-deployment-9456bbbf9-2zrxr to k8s-work
er
  Normal  Pulled     62s   kubelet            Container image "nginx:1.14.2" already present on machine
  Normal  Created    62s   kubelet            Created container nginx
  Normal  Started    61s   kubelet            Started container nginx
```

Update a Deployment

Update the nginx Pods to use the nginx:1.16.1 image instead of the nginx:1.14.2 image

>kubectl set image deployment/nginx-deployment nginx=nginx:1.16.1

```
chen@k8s-master:~/nginx$ kubectl set image deployment/nginx-deployment nginx=nginx:1.16.1
deployment.apps/nginx-deployment image updated
```

Alternatively, modify nginx_deployment.yaml file and reconfigure

```
chen@k8s-master:~/nginx$ kubectl apply -f nginx_deployment.yaml
deployment.apps/nginx-deployment configured
```

```
chen@k8s-master:~/nginx$ kubectl get pod -l app=nginx
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-ff6655784-tph2x    1/1     Running   0          9s
nginx-deployment-ff6655784-vcjfk    1/1     Running   0          7s
```

```
Containers:
  nginx:
    Container ID:   docker://3ada746d70ee10882f0c60d2d0675f9f055e92d3191ee58af29
    Image:          nginx:1.16.1
    Image ID:       docker-pullable://nginx@sha256:d20aa6d1cae56fd17cd458f4807e0
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Fri, 22 Apr 2022 09:53:26 +0800
    Ready:          True
    Restart Count:  0
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.16.1
        ports:
        - containerPort: 80
```

Update a Deployment

Scale the Deployment to 4 Pods (replicas=4)

>kubectl scale deployment/nginx-deployment --replicas=4

```
chen@k8s-master:~/nginx$ kubectl scale deployment/nginx-deployment --replicas=4
deployment.apps/nginx-deployment scaled
```

Alternatively, modify nginx_deployment.yaml file and reconfigure

```
chen@k8s-master:~/nginx$ kubectl apply -f nginx_deployment.yaml
deployment.apps/nginx-deployment configured
```

```
chen@k8s-master: /nginx$ kubectl get pod -l app=nginx
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-ff6655784-jgczt    1/1     Running   0          6s
nginx-deployment-ff6655784-jvr9s    1/1     Running   0          6s
nginx-deployment-ff6655784-tph2x    1/1     Running   0          67s
nginx-deployment-ff6655784-vcjfk    1/1     Running   0          65s
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 4
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.16.1
        ports:
        - containerPort: 80
```

Delete a Deployment

Delete the deployment

>kubectl delete deployment nginx-deployment

```
chen@k8s-master:~/nginx$ kubectl delete deployment nginx-deployment
deployment.apps "nginx-deployment" deleted
```

感谢聆听

饮 水 思 源　爱 国 荣 校