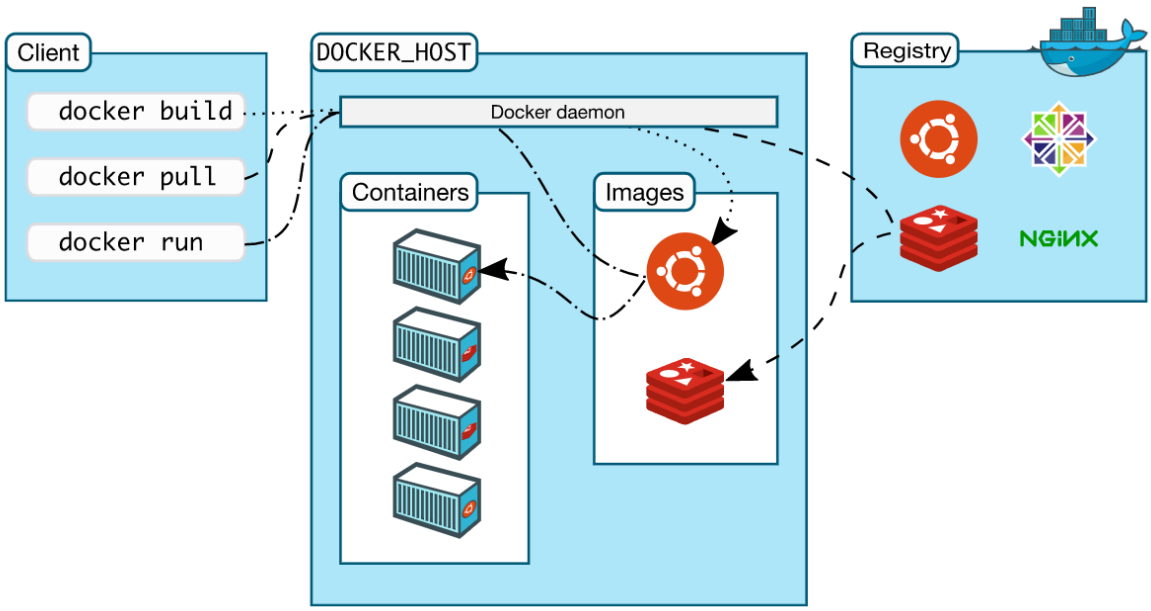
Docker 安装与使用

Docker 简介

> Docker is an open platform for developing, shipping, and running applications. 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.



镜像(Image)

容器(Container)

仓库(Repository)

Docker 安装

```
yum update -y
yum install -y yum-utils
yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
yum install -y docker-ce
docker -v
```

```
[root@iZ2zeewce1gkyyw1ttadd5Z gitlab]# docker -v
Docker version 20.10.4, build d3cb89e
```

```
# 启动docker服务
systemctl start docker
# 开机启动docker服务
systemctl enable docker
# 更换docker阿里源
# https://cr.console.aliyun.com/cn-hangzhou/instances/mirrors 获取源地址
```

```
sudo mkdir -p /etc/docker
sudo tee /etc/docker/daemon.json <<-'EOF'
{
  "registry-mirrors": ["https://yb0l33ts.mirror.aliyuncs.com"]
}
EOF
sudo systemctl daemon-reload
sudo systemctl restart docker
```

Docker Compose 安装

服务编排，按照一定的业务规则批量管理容器

> Docker Compose 是一个编排多容器分布式部署的工具，提供命令集管理容器化应用的完整开发周期，包括服务构建，启动和停止。

```
sudo curl -L "https://github.com/docker/compose/releases/download/1.28.4/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
docker-compose --version
```

```
[root@iZ2zeewce1gkyyw1ttadd5Z gitlab]# docker-compose --version
docker-compose version 1.28.4, build cabd5cfb
```

使用步骤：

1. 利用Dockerfile定义运行环境镜像
2. 使用docker-compose.yml定义组成应用的各服务
3. 运行docker-compose up启动应用

Docker 命令

Docker 服务相关命令

```
# 启动docker服务
systemctl start docker
# 停止docker服务
systemctl stop docker
# 重启docker服务
systemctl restart docker
# 查看docker服务
systemctl status docker
# 开机启动docker服务
systemctl enable docker
```

Docker 镜像相关命令

```
# 查看镜像
docker images
# 查看镜像id
docker images -q
# 搜索镜像
docker search redis
# 拉取镜像
docker pull redis
docker pull redis:6.0
# 删除镜像
docker rmi 镜像id
docker rmi redis:6.0
```

Docker 容器相关命令

```
# 查看正在运行的容器
docker ps
# 查看所有容器
docker ps -a
# 创建容器,后台运行
docker run -id --name=别名 镜像名
# 进入正在运行的容器
docker exec -it 容器id /bin/bash
# 启动容器
docker start 容器名或id
# 停止容器
docker stop 容器名或id
# 重启容器
docker restart 容器名或id
# 删除容器
docker rm 容器名或id
# 查看容器信息
docker inspect 容器名或id
```

Docker 容器的数据卷

数据卷

- 数据卷是宿主机中的一个目录或文件
- 当容器目录和数据卷目录绑定后，对方的修改会立即同步
- 一个数据卷可以被多个容器同时挂载
- 一个窗口也可以被挂载多个数据卷

数据卷作用

- 容器数据持久化
- 外部机器和容器间接通信
- 容器之间数据交换

配置数据卷

创建启动容器，使用-v参数设置数据卷

- docker run ... -v 宿主机目录(文件):容器内目录(文件) ...
- 目录必须是绝对路径
- 如果目录不存在，会自动创建
- 可以挂载多个数据卷

```
dokcer run -it --name=m1 -v /root/data:/root/data_container mysql  /bin/bash
```

数据卷容器

多容器进行数据交换

- 多个容器挂载同一个数据卷
- 数据卷容器

配置数据卷容器

- 创建启动c3数据卷容器，使用-v参数 设置数据卷

```
docker run -it --name=c3 -v /volume mysql /bin/bash
```

- 创建忘却c1 c2 容器，使用--volumes-from 参数设置数据卷

```
docker run -it --name=c1 --volumes-from c3 mysql /bin/bash
docker run -it --name=c2 --volumes-from c3 mysql /bin/bash
```

Docker 应用部署

MySQL 部署

```
# 创建文件夹
mkdir mysql
# 拉取容器
docker pull mysql
# 创建运行容器
docker run -id \
    -p 3306:3306 \
    --name=mysql \
    -v $PWD/conf:/etc/mysql/conf.d \
    -v $PWD/logs:/logs \
    -v $PWD/data:/var/lib/mysql \
    -e MYSQL_ROOT_PASSWORD=123456 \
    mysql
```

Tomcat 部署

```
# 创建文件夹
mkdir tomcat
# 拉取镜像
docker pull tomcat
# 创建运行容器
docker run -id \
    -p 8080:8080 \
    --name=tomcat \
    -v $PWD:/usr/local/tomcat/webapps \
    tomcat
# 将页面放在tomcat目录中即可访问
```

Redis 部署

```
# 拉取容器
docker pull docker
# 创建运行容器
docker run -id --name=redis -p 6379:6379 redis --requirepass "123456"
```

RabbitMq 部署

```
# 拉取容器
docker pull rabbitmq:management
# 创建运行容器
docker run -d -p 5672:5672 -p 15672:15672 --name rabbitmq rabbitmq:management
```

Nginx 部署

```
# 创建文件夹
mkdir nginx
# 创建nginx.conf
cd nginx
mkdir nginx.conf
vim nginx.conf
```

```
user  nginx;
worker_processes  1;

error_log  /var/log/nginx/error.log  warn;
pid        /var/run/nginx.pid;

events {
    worker_connections  1024;
}


http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
```

```
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile        on;
     #tcp_nopush     on;

    keepalive_timeout  65;

    #gzip  on;

    include /etc/nginx/conf.d/*.conf;
}
```

```
# 拉取容器
docker pull nginx
# 创建运行容器
docker run -id \
    --name=nginx \
    -p 80:80 \
    -v $PWD/conf/nginx.conf:/etc/nginx/nginx.conf \
    -v $PWD/logs:/var/log/nginx \
    -v $PWD/html:/usr/share/nginx/html \
    nginx
# 在html文件夹添加index.html
# 访问宿主机ip, index.html 页面显示
```