Docker

安装Dokcer

1. 卸载原有的docker环境

```
yum remove docker \
                  docker-client \
                  docker-client-latest \
                  docker-common \
                  docker-latest \
                  docker-latest-logrotate \
                  docker-logrotate \
                  docker-engine
```

2. 安装需要的安装包

```
yum install -y yum-utils
#更新与yum索引
yum majecache fast
```

3. 设置镜像仓库

```
yum-config-manager \
    --add-repo \
    https://download.docker.com/linux/centos/docker-ce.repo
#阿里云镜像
yum-config-manager \
    --add-repo \
    http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
```

4. 安装docker

```
yum install docker-ce docker-ce-cli containerd.io
```

5. 启动docker

```
systemctl start docker
```

systemctl:Linux的进程管理命令

配置镜像加速器

https://cr.console.aliyun.com/cn-hangzhou/instances/mirrors登陆之后就可以看到了

Docker常用命令

```
docker version
```

查看服务器的Docker信息


```
docker run 镜像名
```

获取仓库中的镜像

```
docker pull 镜像名
```

运行一个镜像

```
docker images
```

查看本地的镜像

```
docker images
```

卸载Docker

```
#卸载依赖
yum remove docker ce docker ce cli contained.io
#删除资源 /var/lib/docker docker的工作路径!
rm -rf /var/lib/docker
```

docker的镜像命令

```
docker info
docker --help
#相当于本地版的Docker官方文档,可以查看相关命令的使用方式
#https://docs.docker.com/engine/reference/commandline/
```

```
docker images
#查看所有本地的主机上的镜像
docker images --help
#查看docker images命令的常用参数
REPOSITORY 镜像的仓库源
TAG 镜像的标签
IMAGE ID 镜像的id
CREATED 镜像的创建时间
SIZE 镜像的大小
#可选项
    -Options:
  -a, --all             Show all images (default hides intermediate images)
      --digests         Show digests
  -f, --filter filter   Filter output based on conditions provided
      --format string   Pretty-print images using a Go template
      --no-trunc        Don't truncate output
  -q, --quiet           Only show image IDs
#搜寻远程仓库中的镜像
docker search <镜像名>
#可选项
  -f, --filter filter   Filter output based on conditions provided
      --format string   Pretty-print search using a Go template
      --limit int       Max number of search results (default 25)
      --no-trunc        Don't truncate output
#搜索star超过5000的镜像
docker search mysql --filter=STARS=5000
```

```
#从远程仓库拉取镜像
docker pull <镜像名>[:tag]
#可选项
    Options:
  -a, --all-tags                Download all tagged images in the repository
      --disable-content-trust   Skip image verification (default true)
      --platform string         Set platform if server is multi-platform capable
  -q, --quiet                   Suppress verbose output
#下载mysql镜像,不加入版本默认使用latest最新版
docker pull mysql
```

```
Using default tag: latest #使用的版本
latest: Pulling from library/mysql
a10c77af2613: Pull complete
b76a7eb51ffd: Pull complete
258223f927e4: Pull complete
2d2c75386df9: Pull complete
63e92e4046c9: Pull complete
f5845c731544: Pull complete
bd0401123a9b: Pull complete
3ef07ec35f1a: Pull complete
```

```
c93a31315089: Pull complete
3349ed800d44: Pull complete
6d01857ca4c1: Pull complete
4cc13890eda8: Pull complete
Digest: sha256:aeecae58035f3868bf4f00e5fc623630d8b438db9d05f4d8c6538deb14d4c31b
#签名
Status: Downloaded newer image for mysql:latest
docker.io/library/mysql:latest #镜像的远程仓库地址
#docker分层下载
#docker指定版本下载
docker pull mysql:5.7
#分层下载
5.7: Pulling from library/mysql
a10c77af2613: Already exists
b76a7eb51ffd: Already exists
258223f927e4: Already exists
2d2c75386df9: Already exists
63e92e4046c9: Already exists
f5845c731544: Already exists
bd0401123a9b: Already exists
2724b2da64fd: Pull complete
d10a7e9e325c: Pull complete
1c5fd9c3683d: Pull complete
2e35f83a12e9: Pull complete
Digest: sha256:7a3a7b7a29e6fbff433c339fc52245435fa2c308586481f2f92ab1df239d6a29
Status: Downloaded newer image for mysql:5.7
docker.io/library/mysql:5.7
```

```
#删除镜像
docker rmi <镜像名字/镜像ID>
#递归删除
docker rmi -f $(docker images -aq)
#docker images -aq先查询出所有的镜像ID,全部删除
```

容器命令

```
#先下载centos的镜像
dokcer pull centos
#启动一个镜像
docker run centos
#选项
docker run --help
#常用的
docker run -d --name nginx01 -p 3304: 80 nginx
--name="Name" 设置容器的名字，用来区分容器
-d 后台方式运行
-it 使用交互方式运行，进入容器查看内容
-p  指定容器的端口(可采用端口映射) -p 8080:8080
-P  随即指定端口
#docker run -it centos /bin/bash
#exit退出容器内部（会停止容器）
#ctrl p q 退出容器不会停止容器的运行
#列出运行的容器
docker ps
#列出容器（有没有在运行的都会被列出来）、
docker ps -a
```

```
docker ps -a -n=x
#显示出最近的x个容器
docker ps -q#只显示出容器的ID
#删除容器（必须先停止也可以rm -f）
docker stop 容器ID
docker rm 容器ID
#重启容器
docker restart 容器ID
#启动容器，指的是历史容器（也就是现在没有在运行的容器）
docker start 容器ID
```

常用其他命令

```
#后台启动容器(-c shell脚本)
docker run -d centos /bin/sh -c "while true:do echo viking;sleep 1;done"
#但是如果没有前台进程的话，docker就会自动停止
#查看日志的命令
docker logs -tf --tail 10 <容器ID>
#查看docker容器内部的进程信息
docker top <容器ID>
#查看容器元数据
docker inspect <容器ID>
```


```
{
    "Id": "d28bdb34c461508afc4336116fbf1960a335838f3d721fa9aeb50cf5c8a36661",
    "Created": "2021-11-23T10:11:14.956740842Z",
    "Path": "docker-entrypoint.sh",
    "Args": [
        "redis-server"
    ],
    "State": {
        "Status": "running",
        "Running": true,
        "Paused": false,
        "Restarting": false,
        "OOMKilled": false,
        "Dead": false,
        "Pid": 3168,
        "ExitCode": 0,
        "Error": "",
        "StartedAt": "2021-11-23T10:11:15.500428527Z",
        "FinishedAt": "0001-01-01T00:00:00Z"
    },
    "Image": "sha256:40c68ed3a4d246b2dd6e59d1b05513accbd2070efb746ec16848adc1b8e07fd4",
    "ResolvConfPath": "/var/lib/docker/containers/d28bdb34c461508afc4336116fbf1960a335838
721fa9aeb50cf5c8a36661/resolv.conf",
    "HostnamePath": "/var/lib/docker/containers/d28bdb34c461508afc4336116fbf1960a335838f3
1fa9aeb50cf5c8a36661/hostname",
    "HostsPath": "/var/lib/docker/containers/d28bdb34c461508afc4336116fbf1960a335838f3d72
9aeb50cf5c8a36661/hosts",
```


```
#进入当前正在运行的容器，开启新的终端（常用）
docker exec -it <容器ID> /bin/bash
#进入容器，不开启新的命令行，不会启动新的进程!
docker attach <容器ID>
```

```
#从容器内拷贝文件到主机上
docker cp 容器id:容器内路径 目的的主机路径
docker cp 92b3c4f524af:/home/test.java /home
#将一个自定义过的容器提交为一个新的镜像
docker commit -a="作者名字" -m="描述" <容器ID> <镜像名>:版本号
#docker查看镜像变更命令
docker history <容器ID>
```

可以选择可视化界面

portainer(不建议使用，黑框框才是程序员的浪漫)

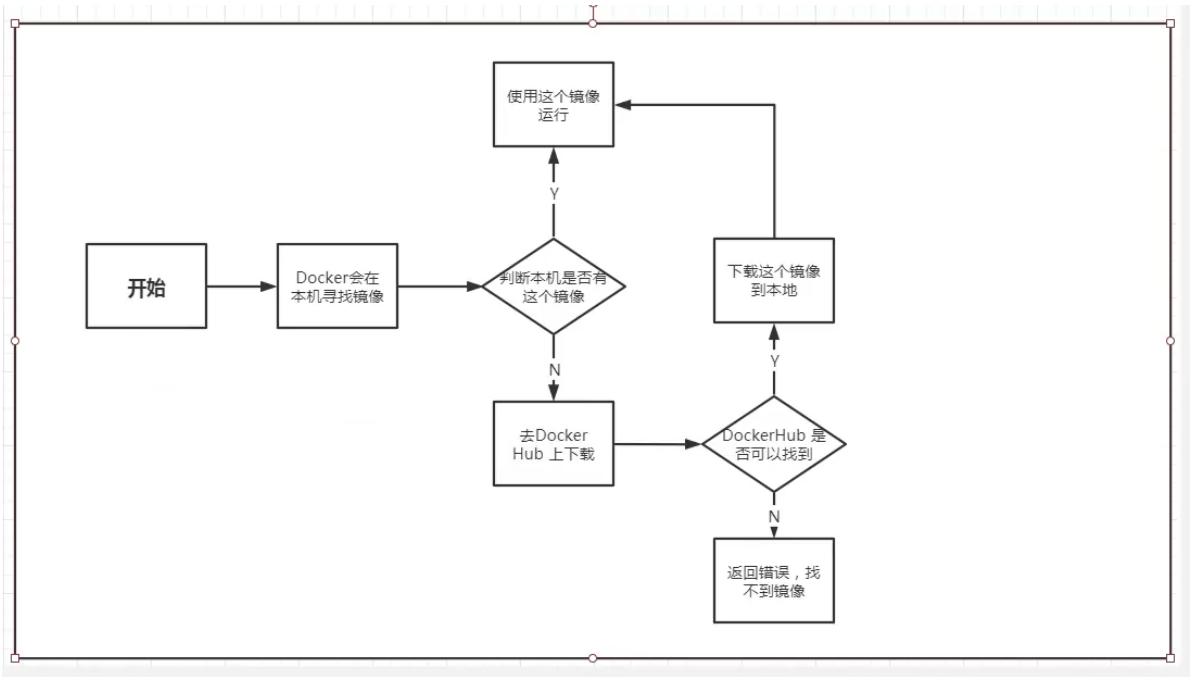
```
docker run -d -p 8088:9000 \
--restart=always -v /var/run/docker.sock:/var/run/docker.sock --privileged=true
portainer/portainer
```

阿里云镜像加速

1. 登陆阿里云
2. 找到镜像加速地址
3. 配置使用docker，直接按照上面的命令敲一遍

Docker原理

Docker运行流程图

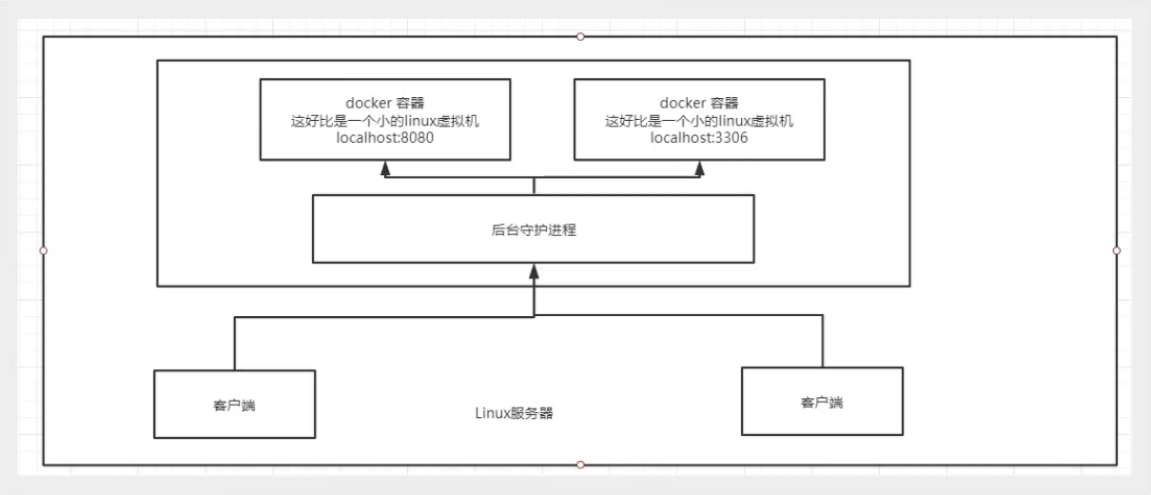


Docker底层原理

Docker是什么工作的？

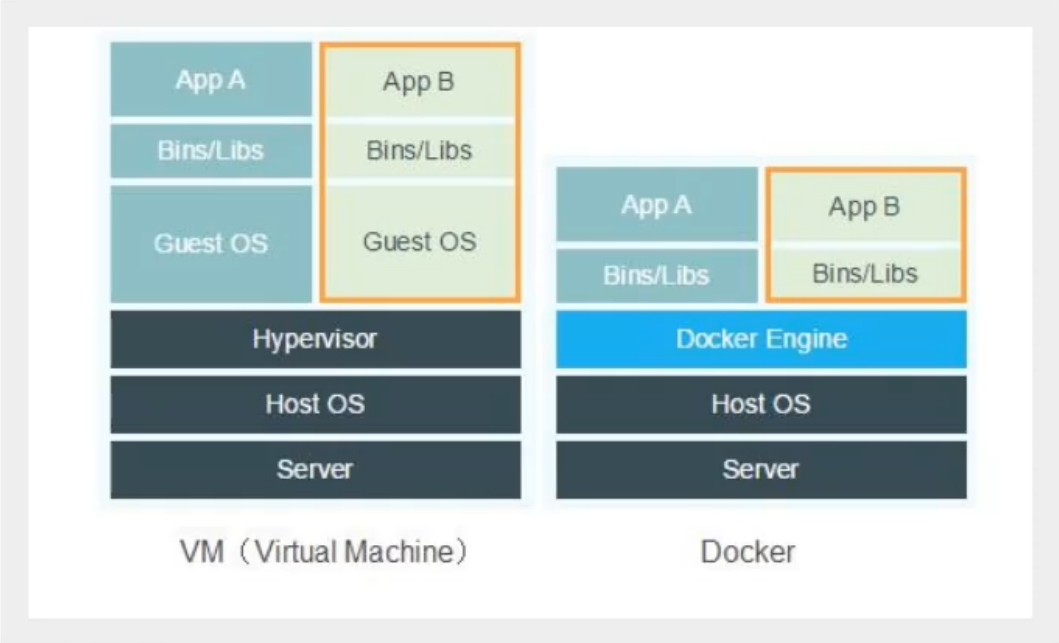
Docker 是一个 Client - Server 结构的系统，Docker的守护进程运行在主机上。 通过Socket从客户端访问！

DockerServer 接收到 Docker-Client 的指令，就会执行这个命令！



Docker为什么比虚拟机快

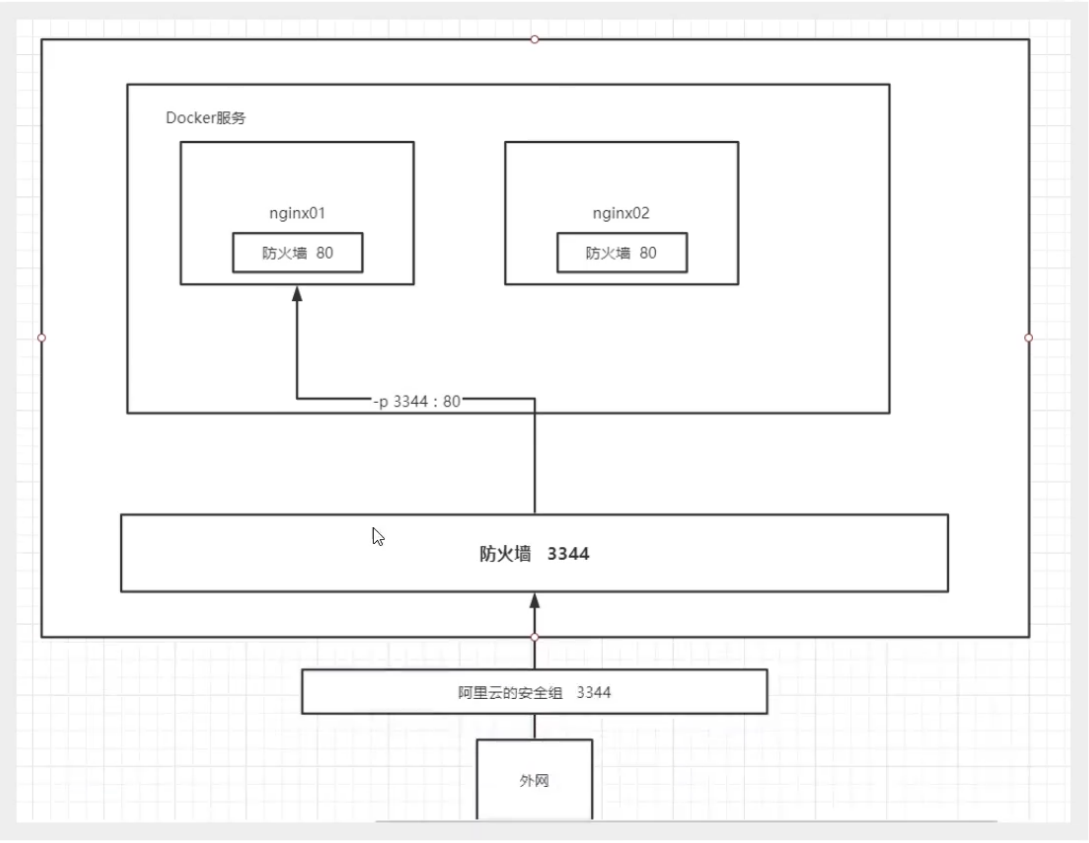
1. Docker有着比虚拟机更少的抽象层



2. docker利用的是宿主机的内核，vm需要用Guest Os；docker在加载一个容器的时候，不需要像虚拟机一样重新加载一个操作系统的内核，避免引导。虚拟机是加载Guest OS，而docker是利用宿主机的操作系统，省略了这个复杂的过程

	Docker容器	LXC	VM
虚拟化类型	OS虚拟化	OS虚拟化	硬件虚拟化
性能	=物理机性能	=物理机性能	5%-20%损耗
隔离性	NS 隔离	NS 隔离	强
QoS	Cgroup 弱	Cgroup 弱	强
安全性	中	差	强
GuestOS	只支持Linux	只支持Linux	全部

3. docker的端口映射



容器数据卷

容器和本地进行数据共享，Docker容器中产生的数据，同步到本地;实即上就是目录的挂载在本地

```
docker run -it -v /home/ceshi1:/home centos /bin/bash
#使用命令挂载，启动起来时可以通过
docker inspect 容器ID
#查看mounts字段
```

实现mysql数据库的数据同步

```
#可以查看官方文档DockerHub的描述，-e加上MYSQL的密码配置
docker run -d -p 5000:3306 -e MYSQL_ROOT_PASSWORD=123456 --name mysql01
mysql:5.7
#可以通过navicat连接服务器的5000，5000和容器内的3306映射，然后本地就可以连接上了
#进入容器
docker exec -it mysql /bin/bash
#启动mysql
mysql -u root -p
#输入密码后创建数据库
show databases;
create database db_account;
use db_account
#创建表
create table user(
    username varchar(25),
    password varchar(25),
    id int(10) primary key
    );
#插入数据
insert into  user values("viking","12345","1");
```

挂载数据卷的两种方式

```
#指定挂载的数据卷即可，不用指定本地的挂载目录
docker run -d -P --name nginx01 -v /etc/nginx nginx
#查看所有数据卷情况（可以拿到数据卷的名字，匿名挂载的话类似一个ID）
docker volume ls
#具名挂载数据卷,这里的juming-nginx并不是一个目录，因为前面没有/，只是作为这个挂载数据卷的名字
docker run -d -P --name nginx02 -v juming-nginx:/etc/nginx nginx
#查看挂载的具体地址(加上具名的名字或者匿名的ID)
docker volume inspect juming-nginx
#当然docker容器所有的卷，没有指定目录的情况下都是在/var/lib/docker/volumes/xxxx/ data
匿名挂载 -v 容器内路径
具名挂载 -v 卷名：容器内路径
指定路径挂载 -v /宿主机路径：容器内路径
指定权限挂载 -v ...:容器内路径:权限  （权限具体指的是容器内对这个文件的权限）
ro readonly/rw 可读可写
```

数据卷容器

两个或者多个容器实现数据共享

```
#当你使用docker首先把一个容器挂载在本地目录
#具名挂载/etc/volume01，位置应该是默认位置
docker run -it --name docker01 -v volume01:/etc/volume01 centos
#启动一个新的容器，挂载到docker01这个容器上
docker run -it --name docker02 --volumes-from docker01 centos
#这样就会同步volume01这个文件夹的操作，docker01叫做数据卷容器
#哪怕删了docker01，文件和数据不会消失
```

DockerFile

DockerFile就是用来构建docker镜像的构建文件

```
#通过DockerFile挂载
FROM centos
#匿名挂载volume01和volume02
VOLUME ["volume01","volume02"]
CMD echo "----end----"
```

1. 编写一个dockerfile文件
2. docker build 构建成为一个镜像
3. docker run 运行一个镜像
4. docker push 发布镜像（DockerHub，阿里云镜像仓库）

基础知识

1. 每个关键字（指令）都是必须是大写字母
2. 执行从上到下顺序执行
3. #表示注释
4. 每一个指令都会创建提交一个镜像层，并提交！

dockerfile是面向开发的，我们以后要发布项目，做镜像，就需要编写dockerfile文件，这个文件十分简单！

指令

```
FROM    #基础镜像，一切从这里开始构建
MAINTAINER      #镜像作者
RUN     #镜像构建的时候要运行的指令
ADD     #步骤：tomacat镜像，添加内容
WORKDIR     #镜像的工作目录（刚进入容器的默认目录）
VOLUME      #挂载的目录
EXPOSE      #暴露端口配置和-p是一个原理
CMD         #指定这个容器启动的时候要运行的命令，只有最后一个会生效，可以被替代
ENTRYPOINT  #指定这个容器启动的时候要运行的命令，可以追加命令
ONBUILD     #构建一个被继承DockerFile这个时候就会运行ONBUILD的指令，触发指令
COPY        #类似ADD，将我们文件拷贝到镜像中
ENV         #构建的时候设置环境变量
```

```
#利用dockerfile构建镜像-f后面加上文件名-t指定镜像名和版本号.不能忘
docker build -f mydockerfile-centos -t mycentos:0.1 .
```

```
FROM centos

MAINTAINER Viking


ENV MYPATH /usr/local

WORKDIR $MYPATH


RUN yum -y install vim


EXPOSE 80
```

```
CMD /bin/bash
```