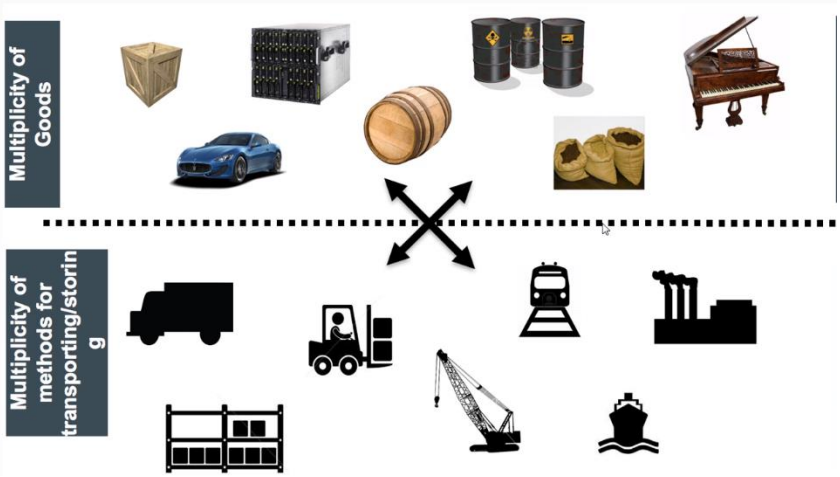
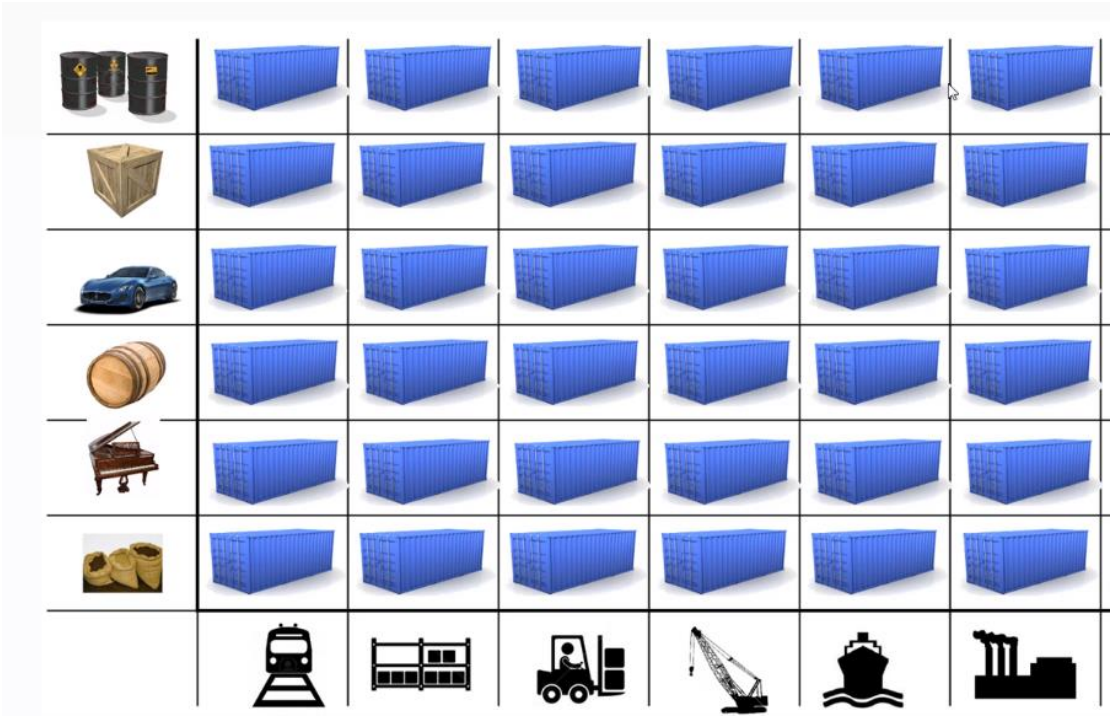
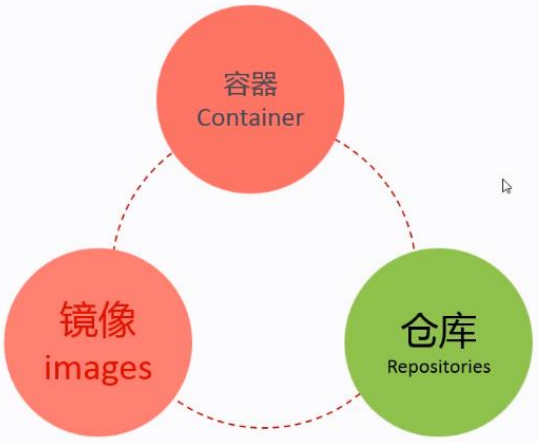
1 docker 介绍

每一次运输，货主与承运方都会担心因货物类型的不同而导致损失，比如几个铁桶错误地压在了一堆香蕉上。另一方面，运输过程中需要使用不同的交通工具也让整个过程痛苦不堪：货物先装上车运到码头，卸货，然后装上船，到岸后又卸下船，再装上火车，到达目的地，最后卸货。一半以上的时间花费在装、卸货上，而且搬上搬下还容易损坏货物。





从技术角度看，传统容器只解决了容器执行（run）问题，而Docker定义了一套容器构建（build）分发（ship）执行（run）



1.1 Docker 是什么

- 官方的说法

Docker 是一个开源的应用容器引擎，让开发者可以打包他们的应用以及依赖包到一个可移植的容器中，然后发布到任何流行的 Linux 机器上，也可以实现虚拟化，容器是完全使用沙箱机制，相互之间不会有任何接口。

- 简单的理解

比喻我们开发一个网站，当不采用 docker，通常要一台物理或虚拟机，然后在上面安装 jdk、mysql、tomcat、自己编写的程序 war 包，这样就完成了一个网站的部署工作；但是这种方式会出现很多问题，如部署复杂、测试生产环境切换经常出问题、各组件之间没有隔离，当采用多个虚拟机时系统较笨重、启动慢、开销大等问题。而 docker 就是为了解决这些问题，为各种应用提供一个运行环境，并且方便高效，相当于一个简化版的 linux。

1.2 Docker 做什么

（1） Docker 是一个便携的应用容器，可以比作一个轻量级的虚拟机；
（2） 容器中可以运行数据库、node.js、api 服务器等；
（3） 搭建环境非常简单，如运行 mysql,一句命令即可：docker run -d -p 3306:3306 tutum/mysql；
（4） 非常适合各种演示环境的搭建，不必任何配置工作；
（5） 保证运行效果在任何机器上一致，不会出现本地可以运行，远程不能运行的情况；
（6） 大多数云主机全面支持 docker，不需要本地部署一套环境，还要在云主机上部署一套环境；

1.3 Docker 原理

基于 LXC 的高级容器引擎，Docker 在 LXC 的基础上进行了进一步的封装，使得操作理加简单，什么又是 linux 的 LXC 技术呢？介绍较全面的文章：https://draveness.me/docker

- 如果我们在服务器上启动了多个服务，这些服务其实会相互影响的，每一个服务都能看到其他服务的进程，也可以访问宿主机器上的任意文件，这是很多时候我们都不愿意看到的，我们更希望运行在同一台机器上的不同服务能做到**完全隔离**，就像运行在多台不同的机器上一样；

- 一旦服务器上的某一个服务被入侵，那么入侵者就能够访问当前机器上的所有服务和文件，这也是我们不想看到的；

- 设置进程相关的命名空间，还会设置与用户、网络、IPC 以及 UTS 相关的命名空间；

- 容器与镜像的关系类似于面向对象编程中的对象与类；

- 容器可以是一个应用，也可以是一组应用。

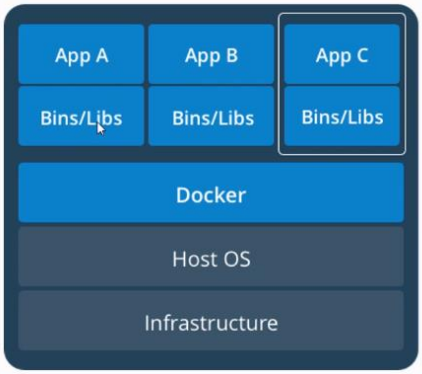
1.4 与虚拟机相比优势

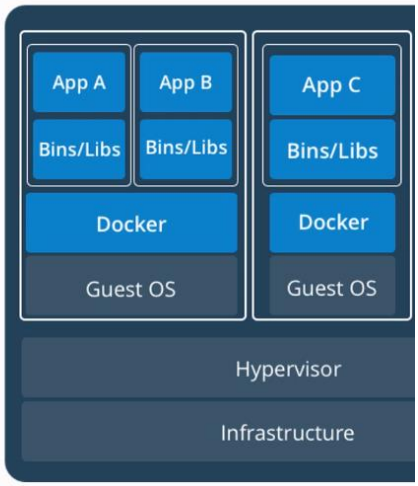
docker 基于当前操作系统虚拟化，直接复用了当前操作系统资源，而虚拟机是基于硬件虚拟化。

特性	容器	虚拟机
启动	秒级	分钟级
硬盘使用	一般为 MB	一般为 GB
性能	接近原生	弱于
系统支持量	单机支持上千个容器	一般几十个

类目	虚拟机	容器
占用磁盘空间	大，G级别	小，M级别
启动速度	慢，分钟级	快，秒级启动
运行形态	运行于Hypervisior上	运行在宿主机内核 共享同一个Linux内核
虚拟化方式	硬件层面实现	操作系统层面 直接复用宿主机操作系统
并发性	单机十几个	成百上千
性能	逊于宿主机	接近宿主机本地进程
资源复用率	低	高
隔离性	单机无法有效隔离	进程级别隔离

- 传统的虚拟化技术，比如 VMWare, KVM, Xen，目标是创建完整的虚拟机。为了运行应用，除了部署应用本身及其依赖（通常几十 MB），还得安装整个操作系统（几 GB）。
- 由于所有的容器共享同一个 Host OS，这使得容器在体积上要比虚拟机小很多。另外，启动容器不需要启动整个操作系统，所以容器部署和启动速度更快，开销更小，也更容易迁移。





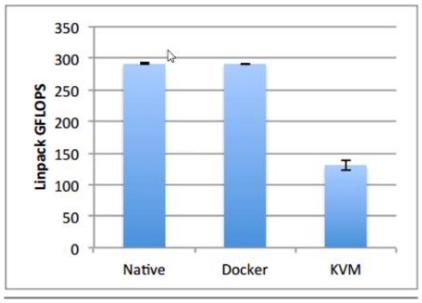
性能对比

以下的数据均是在IBM x3650 M4服务器测得，其主要的硬件参数是：

（1）2颗英特尔xeon E5-2655处理器，主频2.4-3.0 GHz。每颗处理器有8个核，因此总共有16个核。

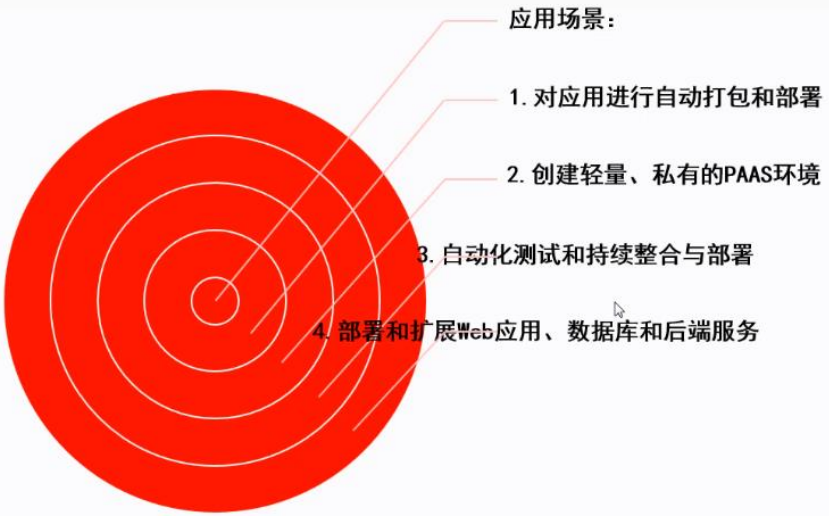
（2）256 GB RAM.

在测试中是通过运算Linpack程序来获得计算能力数据的。结果如下图所示：

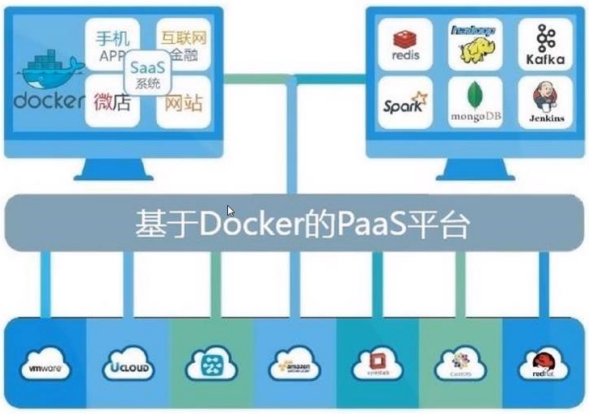


Docker相对虚拟机不足之处

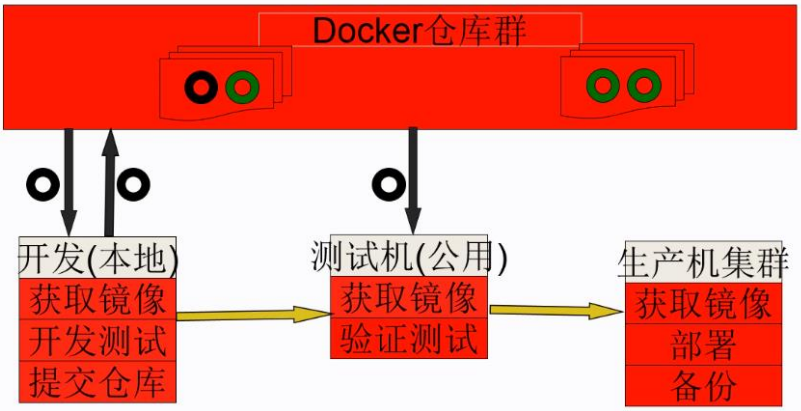
1. 资源隔离方面不如虚拟机，docker是利用cgroup实现资源限制的，只能限制资源消耗的最大值，而不能隔绝其他程序占用自己的资源。
2. 安全性问题。docker目前并不能分辨具体执行指令的用户，只要一个用户拥有执行docker的权限，那么他就可以对docker的容器进行所有操作，不管该容器是否是由该用户创建。比如A和B都拥有执行docker的权限，由于docker的server端并不会具体判断docker client是由哪个用户发起的，A可以删除B创建的容器，存在一定的安全风险。
3. docker目前还在版本的快速更新中，细节功能调整比较大。一些核心模块依赖于高版本内核，存在版本兼容问题



Ø 创建轻量、私有的**PAAS**环境
(Creation of lightweight, private PAAS environments)

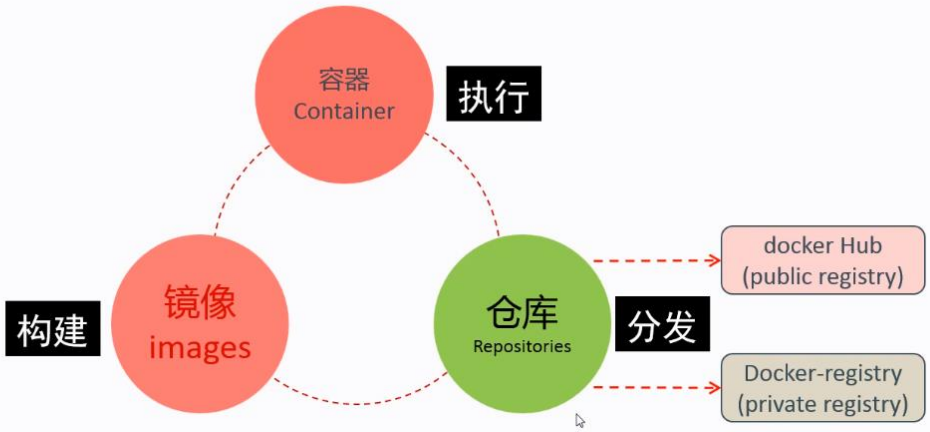


以Docker为单位的开发部署流程设计



- 以docker为单位的开发测试部署流程，简化了环境搭建的步骤，提高了资源利用效率和开发测试部署的速度，降低了迁移的成本
- 更快速的交付和部署。使用Docker，开发人员可以使用镜像来快速构建一套标准的开发环境；开发完成之后，测试和运维人员可以直接使用相同环境来部署代码。
- Docker可以快速创建和删除容器，实现快速迭代，大量节约开发、测试、部署的时间。并且，各个步骤都有明确的配置和操作，整个过程全程可见，使团队更容易理解应用的创建和工作过程。

基本概念（三大核心）

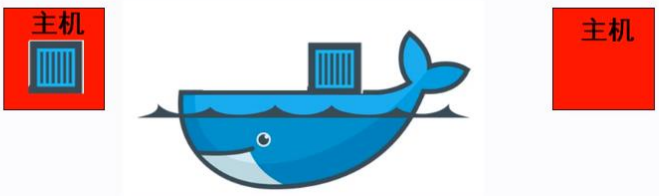


容器

- 等同于从模板中创建虚拟机
- 容器是从镜像创建的运行实例。它可以被启动、开始、停止、删除。每个容器都是相互隔离的、保证安全的平台。
- 可以把容器看做是一个简易版的 Linux 环境（包括root用户权限、进程空间、用户空间和网络空间等）和运行在其中的应用程序。

仓库及仓库注册服务器

- 仓库是集中存放镜像文件的场所
- 仓库注册服务器上往往存放着多个仓库，每个仓库中又包含了多个镜像，每个镜像有不同的标签
- 仓库分为公开仓库（Public）和私有仓库（Private）两种形式
- push 镜像到仓库，从仓库pull下镜像



1.5 理解

- Dock 需要先安装这一软件
- 使用 dock 拉远程镜像，如从阿里云下载镜像
- 使用 dock 命令可以看到下载的镜像列表
- 使用 dock run 下载的镜像，run 时可以指定参数(指定端口，当前机器与镜像文件映射)；
- Run 后就运行起一个实例服务了，如 mysql、redis
- Spring boot 的 jar 包程序，可以打包成镜像，需要在 jar 所在的目录加入配置文件，运行 dock 命令生成镜像，然后 run 这个镜像；
- 对已运行的实例，还可以生成新的镜像（如修改了一些参数后）；
- Run 状态的服务可以查看到 id,通过 id 启动、停止；
- 可以通过 id 进入到镜像内部查看（目录与 linux 的默认目录相似）；

2 Docker 安装

参数网址
http://www.runoob.com/docker/centos-docker-install.html

1. **查看操作系统版本和位数**

```
uname  -r
```

2. **先删除一下 docker**

```
yum remove docker
```

3. **安装一些必要的系统工具**

```
yum install -y yum-utils device-mapper-persistent-data lvm2
```

4. **添加软件源信息，添加 yum 源**

或者直接下载 rpm 安装

```
wget
https://download.docker.com/linux/centos/7/x86_64/stable/Packages/docker-ce-17.09.0.ce-1.el7.centos.x86_64.rpm
yumlocalinstall docker-ce-17.09.0.ce-1.el7.centos.x86_64.rpm

yum-config-manager--add-repo
http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
```

5. **更新 yum 缓存**

```
yummakecache fast
```

6. 安装 Docker-ce

moby 是继承了原先的 docker 的项目，是社区维护的的开源项目，谁都可以在 moby 的基础打造自己的容器产品

docker-ce 是 docker 公司维护的开源项目，是一个基于 moby 项目的免费的容器产品

docker-ee 是 docker 公司维护的闭源产品，是 docker 公司的商业产品

```
yum -y install docker-ce
```

7. 启动 Docker 后台服务

```
systemctl start docker
```

8. 开机自启

```
systemctl enable docker
```

9. 拉取 hello-world 镜像

```
docker pull hello-world
docker images
```

10. 测试

输入下列命令后，如出现“Hello from Docker!”则说明运行成功

```
docker run hello-world
dockerps - a
```

11. 查看 docker 信息

显示版本、镜像数量、容器数量

```
docker info
```

3 镜像操作

1. 镜像查询

从公网查询 mysql 镜像

```
docker searchhell-world
docker search mysql
```

2. 拉取查询到本地

```
docker pull ubuntu:13.10
docker pull registry.cn-hangzhou.aliyuncs.com/acs-sample/mysql:5.7
```

3. 查看镜像信息

如查看 hell-world 镜像信息
docker inspect hello-world

4. 运行镜像
运行镜像后就会产生一个实例，即一个容器
运行 hello-world 镜像
docker run hello-world
运行 mysql 镜像
参数说明：-P:暴露所有端口（映身出来的端口是随机的），-p:容器端口和机器端口映射（端口是确定的），下方的-p 3306（宿主机端口）:3306(容器端口)；
docker run --name mysqlserver -v $PWD/conf:/etc/mysql/conf.d -v $PWD/logs:/logs -v $PWD/data:/var/lib/mysq -e MYSQL_ROOT_PASSWORD=123456 -d -i -p 3306:3306 mysql:5.7

5. 修改镜像名称
docker tag

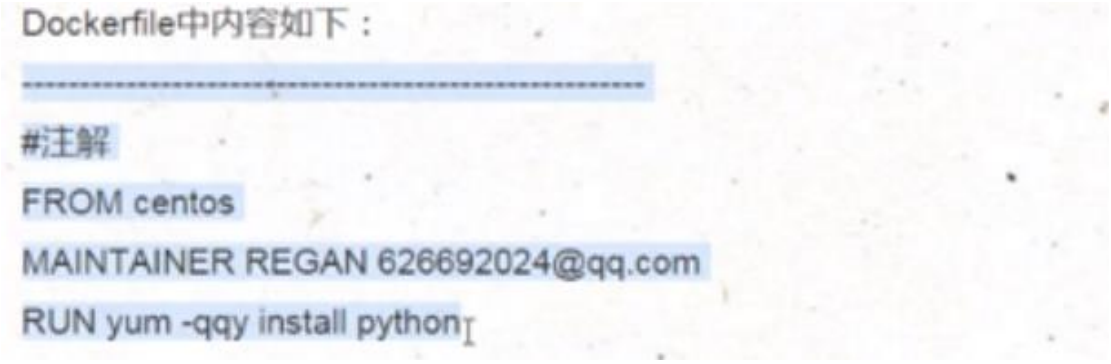
6. 以文件方式导出、载入镜像(当没有本地创库时可采用此方式)
把镜像导出为文件
docker save -o /opt/hello-world.tar hello-world
把镜像导出为文件
docker load --input /opt/hello-world.tar

7. 创建镜像
（1） 通过 commit 方式创建镜像
docker commit -m "my hello-world" -a "test hello-world" 811d19f761cf test-hell-world
镜像创建后，就可以通过 docker images 查看了
docker images 5725cod57443 my/centos:v1.1

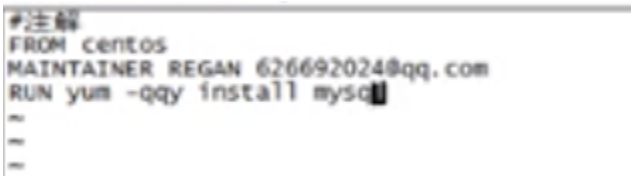
（2） Dockerfile 创建镜像
commit 虽然容易扩展，但不便于团队的分享，
- 创建一个空白文件夹

mkdircentos_withmysql
- 在该目录下，创建一个 Dockerfile 文件，名称不能搞错：

添加 python



添加 mysql



- 创建镜像

其中.表示当前目录

- 查看

docker images

- 将本地镜像上传到公有云仓库

因为标签名称，必须以 github 上的账号为名称，然后再 push 上 github 上

- 删除所有镜像

docker rmi $(docker images)

4 容器操作

```
dockerps-a// 查看所有容器
dockerps// 查看所有正在运行容器
```

docker stop 3864402de766 //停止容器,3864402de766 为容器 ID
dockerrm 3864402de766 //删除容器

```
dockerps-a-q// 查看所有容器 ID
docker stop $(docker ps -a -q) //  stop 停止所有容器
docker rm $(docker ps -a -q) //   remove 删除所有容器
```

docker pause //暂停某一容器的所有进程
docker update --restart=always 7c20426dcc5a(容器 ID) //设置容器开机自启

1. 进行到容器内部

docker exec –ti myhbase /bin/bash
进入到容器内部，可以查看目录结构，可以像操作虚拟机一样进行各种操作，输入“exit”则退出容器，回到宿主机

2. 导入、导出容器

导出为一个文件
docker export myhbase>/opt/myhbase.tar
导入为一个镜像

cat /opt/myhbase.tar | sudodocker import - myhbase/2.2.0
查看效果
docker images

3. 查看容器日志
dockerlogs 3864402de766

4. 使用 run 命令执行一个 docker 容器后，停止后使用 start 启动后，原来的 run 参数仍然生效，如下：
docker run -p 8200:8200 sinhic/sinhic-eureka
docker start 1b0e98211419
发现使用 start 启动后，8200 端口也会映射出来

5. 从容器中拷出文件
docker cp testtomcat：/usr/local/tomcat/webapps/test/js/test.js /opt

4.1 查看容器日志

docker logs [OPTIONS] CONTAINER Options: --details 显示更多的信息 -f, --follow 跟踪实时日志 --since string 显示自某个 timestamp 之后的日志，或相对时间，如 42m（即 42 分钟） --tail string 从日志末尾显示多少行日志，默认是 all -t, --timestamps 显示时间戳 --until string 显示自某个 timestamp 之前的日志，或相对时间，如 42m（即 42 分钟）

- 查看最近 20 行日志

docker logs --tail=20 83e92f14f978

- **查看全部日志**

docker logs -f e363cedffca2

5 进阶（高级部分）

5.1 docker 目录

```
[root@master ~]# find / -name docker
/run/docker
/sys/fs/cgroup/memory/docker
/sys/fs/cgroup/hugetlb/docker
/sys/fs/cgroup/perf_event/docker
/sys/fs/cgroup/freezer/docker
/sys/fs/cgroup/devices/docker
/sys/fs/cgroup/cpuset/docker
/sys/fs/cgroup/cpu,cpuacct/docker
/sys/fs/cgroup/blkio/docker
/sys/fs/cgroup/net_cls/docker
/sys/fs/cgroup/systemd/docker
/etc/docker
/var/lib/docker
/usr/bin/docker
/usr/share/bash-completion/completions/docker
```

我们看到 /sys/fs/cgroup 目录中有若干个子目录，我们可以认为这些都是受 cgroups 控制的资源以及这些资源的信息。

- blkio — 这个子系统为块设备设定输入/输出限制，比如物理设备（磁盘，固态硬盘，USB 等等）。
- cpu — 这个子系统使用调度程序提供对 CPU 的 cgroup 任务访问。
- cpuacct — 这个子系统自动生成 cgroup 中任务所使用的 CPU 报告。
- cpuset — 这个子系统为 cgroup 中的任务分配独立 CPU（在多核系统）和内存节点。
- devices — 这个子系统可允许或者拒绝 cgroup 中的任务访问设备。
- freezer — 这个子系统挂起或者恢复 cgroup 中的任务。
- memory — 这个子系统设定 cgroup 中任务使用的内存限制，并自动生成内存资源使用报告。
- net_cls — 这个子系统使用等级识别符（classid）标记网络数据包，可允许 Linux 流量控制程序（tc）识别从具体 cgroup 中生成的数据包。
- net_prio — 这个子系统用来设计网络流量的优先级
- hugetlb — 这个子系统主要针对于 HugeTLB 系统进行限制，这是一个大页文件系统。

https://blog.csdn.net/u010884123/article/details/60593405

1、/var/lib/docker/devicemapper/devicemapper/data #用来存储相关的存储池数据
2、/var/lib/docker/devicemapper/devicemapper/metadata #用来存储相关的元数据。
3、/var/lib/docker/devicemapper/metadata/ #用来存储 device_id、大小、以及传输_id、初始化信息
4、/var/lib/docker/devicemapper/mnt #用来存储挂载信息
5、/var/lib/docker/container/ #用来存储容器信息
6、/var/lib/docker/graph/ #用来存储镜像中间件及本身详

细信息和大小 、以及依赖信息

7、/var/lib/docker/repositores-devicemapper #用来存储镜像基本信息

8、/var/lib/docker/tmp #docker 临时目录

9、/var/lib/docker/trust #docker 信任目录

10、/var/lib/docker/volumes #docker 卷目录

查看 docker 所有目录信息

```
find / -name docker
```

1. 修改镜像默认路径

- Docker 默认的镜像和容器存储位置在/var/lib/docker

```
cd /etc/systemd/system/multi-user.target.wants
vimdocker.service
```

--graph=/data/docker：docker 新的存储位置

--storage-driver=overlay ： 当前 docker 所使用的存储驱动

ExecStart=/usr/bin/dockerd --graph=/data/docker --storage-driver=overlay --registry-mirror=https://jxus37ad.mirror.aliyuncs.com

- 重启一下：

```
systemctl daemon-reload
systemctl restart docker
```

5.2 镜像信息

5.3 run 镜像

https://docs.docker.com/engine/reference/commandline/run/

https://www.cnblogs.com/yfalcon/p/9044246.html

常用选项说明

- -d, --detach=false，指定容器运行于前台还是后台，默认为 false

```
docker run -d --restart=always -p 8200:8200 ${imageName}:${tag}
```

- -i, --interactive=false，打开 STDIN，用于控制台交互
- -t, --tty=false，分配 tty 设备，该可以支持终端登录，默认为 false
- -u, --user=""，指定容器的用户
- -a, --attach=[]，登录容器（必须是以 docker run -d 启动的容器）
- -w, --workdir=""，指定容器的工作目录

-w 参数覆盖掉 WORKDIR 指令的设置，如：

执行 docker run -w / myimage

上面的-w 参数将容器的工作目录设置成了根目录

指定工作目录，与 dockerfile 指令 WORKDIR 相对应，如果没有指定，则工作目录是 "/"，即根目录

- -c, --cpu-shares=0，设置容器 CPU 权重，在 CPU 共享场景使用
- -e, --env=[]，指定环境变量，容器中可以使用该环境变量

```
docker run --name mysqlserver -v $PWD/conf:/etc/mysql/conf.d -v $PWD/logs:/logs -v $PWD/data:/var/lib/mysq -e MYSQL_ROOT_PASSWORD=123456 -d -i -p 3306:3306 mysql:5.7
```

- -m, --memory=""，指定容器的内存上限
- -P, --publish-all=false，指定容器暴露的端口

- -p, --publish=[]，指定容器暴露的端口

```
docker run -d --restart=always -p 8200:8200 ${imageName}:${tag}
```

第一个 8200 是宿主机的端口，第二个 8200 是容器自己的端口，可以理解为你要访问宿主机，那么就会转到容器的某个端口，而且容器要暴露这个端口才行。

- -h, --hostname=""，指定容器的主机名
- -v, --volume=[]，给容器挂载存储卷，挂载到容器的某个目录

```
docker run --name mysqlserver -v $PWD/conf:/etc/mysql/conf.d -v $PWD/logs:/logs -v $PWD/data:/var/lib/mysq -e MYSQL_ROOT_PASSWORD=123456 -d -i -p 3306:3306 mysql:5.7
```

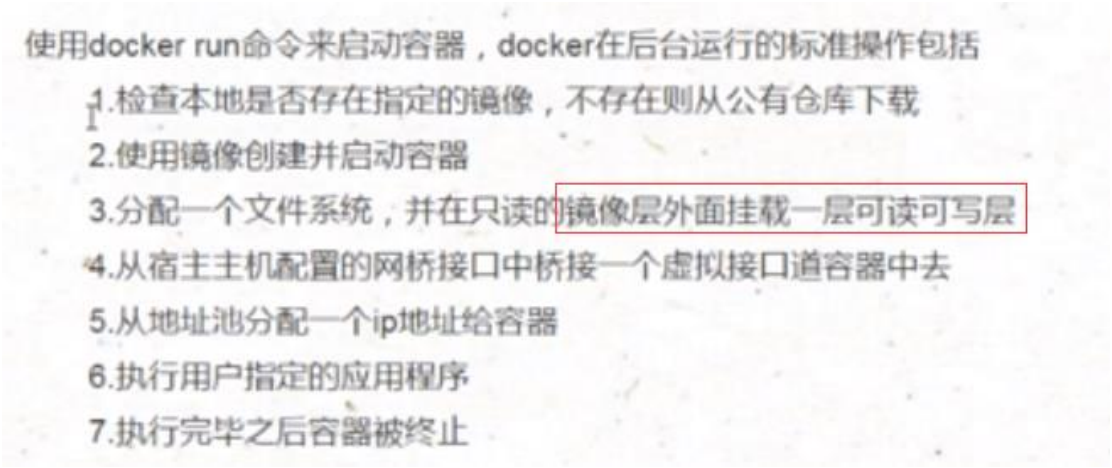
映射容目录目录到宿主机目录

- --volumes-from=[]，给容器挂载其他容器上的卷，挂载到容器的某个目录

- --cap-add=[]，添加权限，权限清单详见：http://linux.die.net/man/7/capabilities
- --cap-drop=[]，删除权限，权限清单详见：http://linux.die.net/man/7/capabilities

- --cidfile=""，运行容器后，在指定文件中写入容器 PID 值，一种典型的监控系统用法
- --cpuset=""，设置容器可以使用哪些 CPU，此参数可以用来容器独占 CPU
- --device=[]，添加主机设备给容器，相当于设备直通
- --dns=[]，指定容器的 dns 服务器
- --dns-search=[]，指定容器的 dns 搜索域名，写入到容器的/etc/resolv.conf 文件
- --entrypoint=""，覆盖 image 的入口点
- --env-file=[]，指定环境变量文件，文件格式为每行一个环境变量
- --expose=[]，指定容器暴露的端口，即修改镜像的暴露端口
- --link=[]，指定容器间的关联，使用其他容器的 IP、env 等信息
- --lxc-conf=[]，指定容器的配置文件，只有在指定--exec-driver=lxc 时使用
- --name=""，指定容器名字，后续可以通过名字进行容器管理，links 特性需要使用名字
- --net="bridge"，容器网络设置:
  - bridge 使用 docker daemon 指定的网桥
  - host //容器使用主机的网络
  - container:NAME_or_ID>//使用其他容器的网路，共享 IP 和 PORT 等网络资源
  - none 容器使用自己的网络（类似--net=bridge），但是不进行配置
- --privileged=false，指定容器是否为特权容器，特权容器拥有所有的 capabilities
- --restart="no"，指定容器停止后的重启策略:
  - no：容器退出时不重启
  - on-failure：容器故障退出（返回值非零）时重启
  - always：容器退出时总是重启
- --rm=false，指定容器停止后自动删除容器(不支持以 docker run -d 启动的容器)
- --sig-proxy=true，设置由代理接受并处理信号，但是 SIGCHLD、SIGSTOP 和 SIGKILL 不能被代理

5.4 docker 生命周期



5.5 端口查看

安装 netstat 工具
yum install net-tools
查看本机端口
netstat -tunlp

5.6 基本操作

docker version //显示 Docker 版本信息
docker info //显示 Docker 系统信息，包括镜像和容器数
systemctl enable docker.service //开机 docker 自启
systemctl status docker.service //查看状态

5.7 镜像

--查找外网公共镜像
https://hub.docker.com/

docker search mysql　//命令查询
dockerrmi { REPOSITORY} //删镜像，如果有容器在使用，则无法删除，会提示错误

5.8 Docker pull 超时

vi /etc/docker/daemon.json
{
　"registry-mirrors": ["https://registry.docker-cn.com"]
}

service docker restart

修改后，发现仍然很慢，改成如下地址后好使
{"insecure-registries":["10.101.43.196:5000"],
"registry-mirrors": ["http://4e70ba5d.m.daocloud.io"]