安装:

yum -y install docker

启动服务:

service docker start（最好用root用户启动,不然启动不了)

拉取一个 Docker 镜像：

docker pull centos:latest

注: cento：lastest 是镜像的名称以及版本(默认为最新)，Docker Daemon 发现本地没有我们需要的镜像，会自动去 Docker Hub 上去下载镜像，下载完成后，该镜像被默认保存到 /var/lib/docker 目录下。

来自〈https://mp.weixin.qq.com/s/x2zf854JJCsUz6DRhMVUTg〉

测试运行:

docker run hello-world

docker run ubuntu:15.10 /bin/echo "Hello world"

- **docker: Docker 的二进制执行文件。**
- **run:与前面的 docker 组合来运行一个容器。**
- **ubuntu:15.10指定要运行的镜像，Docker首先从本地主机上查找镜像是否存在，如果不存在，Docker 就会从镜像仓库 Docker Hub 下载公共镜像。**
- **/bin/echo "Hello world": 在启动的容器里执行的命令**

以上命令完整的意思可以解释为：Docker 以 ubuntu15.10 镜像创建一个新容器，然后在容器里执行 bin/echo "Hello world"，然后输出结果。

运行一个新的mysql容器:

docker run -p 3307:3306 --name mysql3 -e MYSQL_ROOT_PASSWORD=123456 -d mysql

运行一个已有的容器

docker start 容器名/容器id

查看主机下存在多少镜像:

docker images

来自〈https://mp.weixin.qq.com/s/x2zf854JJCsUz6DRhMVUTg〉

查看当前有哪些容器在运行 ：

docker ps -a

来自〈https://mp.weixin.qq.com/s/x2zf854JJCsUz6DRhMVUTg〉

制作镜像:

1. 创建文件 Dockerfile,并写入，Dockerfile 中每一条指令都创建镜像的一层(最多127层)

来自〈https://www.cnblogs.com/lsgxeva/p/8746644.html〉

```
# Docker image for springboot file run

# 基础镜像使用java
FROM java:jdk8

# VOLUME 指定了临时文件目录为/tmp。
# 其效果是在主机 /var/lib/docker 目录下创建了一个临时文件，并链接到容器的/tmp
VOLUME /tmp
# 将jar包添加到容器中并更名为app.jar
ADD onlineSpider-0.0.1-SNAPSHOT.jar app.jar
# 运行jar包
RUN bash -c 'touch /app.jar'
ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar"]
```

1. 执行命令 (注意,有个 . 这个点指当前目录)

docker build -t springboot-demo .

来自〈https://blog.csdn.net/junmoxi/article/details/80861199〉

-t 参数是指定此镜像的tag名

删除镜像(如果该镜像正在被容器使用):

1. 先停止容器:

docker ps

docker stop container_name/container_id

2. 删除容器:

docker rm container_name/container_id

来自〈https://mp.weixin.qq.com/s/x2zf854IJCsUz6DRhMVUTg〉

3. 删除镜像;

docker rmi image_name

来自〈https://mp.weixin.qq.com/s/x2zf854IJCsUz6DRhMVUTg〉

删除悬挂镜像

(那些 〈 none〉:〈 none〉样子的镜像,这个样子的有两种,一种的镜像的父层,(镜像是有多层结构的),还有一个种就是无用的(多次使用docker build和docker pull 创建统一个镜像造成)

来自〈https://www.jb51.net/article/124549.htm〉

)

docker rmi $(docker images -f "dangling=true" -q)

构建本地仓库并使用

1. 下载并启动注册器(本地仓库)

docker run -d -p 5000:5000 --restart=always --name local_registry registry:latest

-d 后台运行

-p 端口映射, 宿主机80端口映射给容器的5000端口

--restart=always 容器意外关闭后, 自动重启 (如果重启docker服务, 带这个参数的, 能自动启动为Up状态, 不带这个的,不会自动启动)

--name 给容器起个名字, 可以根据这个名字去停止/启动/删除容器

来自〈https://www.cnblogs.com/zhouyalei/p/6411614.html〉

这就创建了一个本地仓库,以后就用它了

1. 开放端口

firewall-cmd --zone=public --add-port=5000/tcp --permanent;
firewall-cmd --reload;
firewall-cmd --list-all;

1. 重命名镜像

docker tag 原镜像名:tag 新镜像名:tag
docker tag sprintbootdemo:latest 192.168.142.128/sprintbootdemo:spd1.0.0
相当于拉个版本,既然要上传,就要版本控制一下,tag命令不会删除原镜像(更像是复制并重命名,但是镜像id不变)

1. 推送镜像

docker push 192.168.142.128/sprintbootdemo:spd1.0.0

如果报错了,则执行如下:
vi /etc/sysconfig/docker
来自 <https://www.cnblogs.com/zhouyalei/p/6411614.html>
添加 --insecure-registry 192.168.142.128:5000
然后重启: systemctl restart docker.service
再推送即可;

完整文件:
```
# Modify these options if you want to change the way the docker daemon runs
OPTIONS='--selinux-enabled --log-driver=journald --signature-verification=false --insecure-registry 192.168.142.128:5000'
if [ -z "${DOCKER_CERT_PATH}" ]; then
    DOCKER_CERT_PATH=/etc/docker
fi
```

1. 拉取镜像

(最好是换一个台机器)
docker pull 192.168.142.128/sprintbootdemo:spd1.0.0
如果不成功,也在这个服务器上增加 --insecure-registry 192.168.142.128:5000 (如上)
(这仅仅是项目,还没有数据库)
docker run -p 8998:8998 192.168.142.128/sprintbootdemo:spd1.0.0