docker是一个能够把开发的应用程序自动部署到容器的开源引擎
docker依赖于写时复制模型
docker客户端和服务器：c/s架构 docker客户端连接到守护进程或者服务器
docker镜像：比如msyql数据库的镜像，nginx的镜像等
registry：保存用户构建的镜像，公有私有
镜像是docker生命周期中的构建或打包阶段，容器是启动或执行阶段

1. centos 7 安装docker

yum -y update
设置 Docker CE 资源库:
sudo yum install -y yum-utils
sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
sudo yum makecache fast
安装docker
sudo yum -y install docker-ce

设置mirror
https://lug.ustc.edu.cn/wiki/mirrors/help/docker
新版的 Docker 使用 /etc/docker/daemon.json 来配置 Daemon
在该配置文件中加入（没有该文件的话，请先创建一个）：从阿里的镜像上pull
{
"registry-mirrors": ["https://p8b4jflv.mirror.aliyuncs.com";]
}

开放管理端口映射
管理端口在 /lib/systemd/system/docker.service 文件中
将其中第11行的 ExecStart=/usr/bin/dockerd 替换为：
ExecStart=/usr/bin/dockerd -H tcp://0.0.0.0:2375 -H unix:///var/run/docker.sock -H tcp://0.0.0.0:7654
（此处默认2375为主管理端口，unix:///var/run/docker.sock用于本地管理，7654是备用的端口）
将管理地址写入 /etc/profile
echo 'export DOCKER_HOST=tcp://0.0.0.0:2375' >> /etc/profile
source /etc/profile
启动docker
systemctl daemon-reload && service docker start
systemctl daemon-reload && service docker restart
sudo docker run hello-world

```
[root@izwz9dbnlmwmv1co13jzbqz broom]# sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
ca4f61b1923c: Pull complete
Digest: sha256:66ef312bbac49c39a89aa9bcc3cb4f3c9e7de3788c944158df3ee0176d32b751
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://cloud.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/engine/userguide/
```

docker版本

```
[root@izwz9dbnlmwmv1co13jzbqz broom]# docker -v
Docker version 17.12.0-ce, build c97c6d6
```

2. docker 使用

docker images查看镜像

```
[root@izwz9dbnlmwmv1co13jzbqz broom]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello-world         latest              f2a91732366c        7 weeks ago         1.85kB
```

拉centos镜像
sudo docker pull centos 从阿里镜像pull就很快

```
[root@izwz9dbnlmwmv1co13jzbqz broom]# sudo docker pull centos
Using default tag: latest
latest: Pulling from library/centos
af4b0a2388c6: Pull complete
Digest: sha256:2671f7a3eea36ce43609e9fe7435ade83094291055f1c96d9d1d1d7c0b986a5d
Status: Downloaded newer image for centos:latest
```

查看centos镜像

```
[root@izwz9dbnlmwmv1co13jzbqz broom]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
centos              latest              ff426288ea90        30 hours ago        207MB
hello-world         latest              f2a91732366c        7 weeks ago         1.85kB
```

创建容器：基于镜像centos创建容器，并且进入
sudo docker run -i -t centos /bin/bash
-i 保证容器中的STDIN是开启的
-t 创建容器时分配一个伪终端
exit 退出
查看所有的（-a）/运行的容器 下图显示 centos系统的镜像 已经不再运行了

```
[root@izwz9dbnlmwmv1co13jzbqz broom]# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS
        NAMES
386c2e645152        centos              "/bin/bash"         3 minutes ago       Exited (0) 55 seconds ago
        affectionate_agnesi
9d7ce40d976b        hello-world         "/hello"            24 minutes ago      Exited (0) 24 minutes ago
        hopeful_kepler
3ba074431139        hello-world         "/hello"            29 minutes ago      Exited (0) 29 minutes ago
        confident_tereshkova
```

docker run --name refactor -i -t centos /bin/bash，可以通过–name参数指定这个容器的名称refactor，以后可以该名称代替容器ID使用。名称需要保证唯一性
启动已经停止运行的容器：docker start 容器的名字或ID，可以通过容器的名称重启启动已经停止的容器，同样的使用容器的ID也可以。同样的restart会重启一个容器
连接到运行中的容器：docker attach 容器名字或ID docker attach refactor
删掉容器
docker container rm refactor
docker rm `docker ps -a -q`

3. docker_case1 使用docker构建一个现有的服务

创建守护式运行的容器（长期运行的容器）
docker run --name refactor -d centos /bin/bash -c "while true;do echo hello; sleep 1;done"
查看日志
`docker logs -ft drefactor`
查看进程
`docker top refactor`

工程配置（工程放在服务器上打包啊）

Dockerfile

```
▼ main
  ▼ docker
      Dockerfile
```

```
FROM frolvlad/alpine-oraclejdk8:slim
VOLUME /tmp
ADD docker_spring_boot.jar app.jar
RUN sh -c 'touch /app.jar'
ENV JAVA_OPTS=""
ENTRYPOINT [ "sh", "-c", "java $JAVA_OPTS -Djava.security.egd=file:/dev/./urandom -jar /app.jar” ]
```

- `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`。改步骤是可选的，如果涉及到文件系统的应用就很有必要了。`/tmp`目录用来持久化到 Docker 数据文件夹，因为 Spring Boot 使用的内嵌 Tomcat 容器默认使用`/tmp`作为工作目录
- 项目的 jar 文件作为 “app.jar” 添加到容器的
- `ENTRYPOINT` 执行项目 app.jar。为了缩短 Tomcat 启动时间，添加一个系统属性指向 “/dev/urandom” 作为 Entropy Source

pom.xml

```
<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <java.version>1.8</java.version>
    <!--properties节点中设置docker镜像的前缀“springboot”-->
    <docker.image.prefix>springboot</docker.image.prefix>
</properties>

<dependencies>
        <!--加入maven插件“docker-maven-plugin”-->

        <plugin>
            <groupId>org.apache.maven.plugin</groupId>
            <artifactId>docker-maven-plugin</artifactId>
            <version>0.4.11</version>
            <configuration>
                <imageName>${docker.image.prefix}/${project.artifactId}</imageName>
                <dockerDirectory>src/main/docker</dockerDirectory>
                <resources>
                    <resource>
                        <targetPath>/</targetPath>
                        <directory>${project.build.directory}</directory>
                        <include>${project.build.finalName}.jar</include>
                    </resource>
                </resources>
            </configuration>
        </plugin>
```

```
<!--properties节点中设置docker镜像的前缀“springboot”-->
<docker.image.prefix>springboot</docker.image.prefix>

<plugin>
   <groupId>org.apache.maven.plugin</groupId>
   <artifactId>docker-maven-plugin</artifactId>
   <version>0.4.11</version>
   <configuration>
      <imageName>${docker.image.prefix}/${project.artifactId}</imageName>
      <dockerDirectory>src/main/docker</dockerDirectory>
      <resources>
         <resource>
            <targetPath>/</targetPath>
            <directory>${project.build.directory}</directory>
            <include>${project.build.finalName}.jar</include>
         </resource>
      </resources>
   </configuration>
</plugin>
```

出现问题1：No plugin found for prefix 'docker' in the current project and in the plugin groups

需要maven配置

```
<pluginGroups>
  <!-- pluginGroup
   | Specifies a further group identifier to use for pl
  <pluginGroup>com.your.plugins</pluginGroup>
  -->
    <pluginGroup>com.spotify</pluginGroup>
</pluginGroups>
```

```
<pluginGroup>com.spotify</pluginGroup>
```

打包配置

`mvn install`打包
`jar`包和`Dockerfile`放在同一目录下

```
[root@izwz9dbnlmwmv1co13jzbqz docker]# ls
Dockerfile  docker_spring_boot.jar  smart-classroom-web-backend
```

表示把程序中8080端口映射到8083
docker run -d -p 8083:8080 docker
-d表示后台运行

```
[root@izwz9dbnlmwmv1co13jzbqz broom]# cd docker/
[root@izwz9dbnlmwmv1co13jzbqz docker]# ls
Dockerfile  docker_spring_boot.jar  smart-classroom-web-backend
[root@izwz9dbnlmwmv1co13jzbqz docker]# docker run -d -p 8083:8080 docker
98d164f20b64128d054573933b206813f7bc3ff7ceb55c087c7663dfbd995761
```

访问http://120.79.72.250:8084/sskt/login?username=admin&password=123456成功！

120.79.72.250:8083/sskt/login?username=admin&password=123456

{"code":"0","msg":"success","data":{"judge":0,"user":{"role":{"r_id":1,"p_ids":[1,11,111,112,21,31,311,312,3121,3122,313,314,315,316,317,318,319],"r_name":"超级管理员","p_idsStr":"1,11,111,112,21,31,311,312,3121,3122,313,314,315,316,317,318,319,"},"id":4,"r_id":0,"token":"225","username":"admin"}}}

至此，把一个工程注入容器就结束了
docker ps查看镜像

```
[root@izwz9dbnlmwmv1co13jzbqz docker]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
        NAMES
7e5a0bacd1c2        docker              "sh -c 'java $JAVA_O…"   2 minutes ago       Up 2 minutes        0.0.0.0:8084->8
080/tcp   amazing_darwin
98d164f20b64        docker              "sh -c 'java $JAVA_O…"   23 minutes ago      Up 23 minutes       0.0.0.0:8083->8
080/tcp   cranky_hamilton
00e833fb4563        centos              "/bin/bash -c 'while…"   5 hours ago         Up 5 hours
        refactor
```

停止
docker stop 98d164f20b64
移除
docker rm 98d164f20b64