Docker的应用场景

- Web 应用的自动化打包和发布。
- 自动化测试和持续集成、发布。
- 在服务型环境中部署和调整数据库或其他的后台应用。
- 从头编译或者扩展现有的 OpenShift 或 Cloud Foundry 平台来搭建自己的 PaaS 环境。

Docker 的优点

Docker 是一个用于开发，交付和运行应用程序的开放平台。Docker 使您能够将应用程序与基础架构分开，从而可以快速交付软件。借助 Docker，您可以与管理应用程序相同的方式来管理基础架构。通过利用 Docker 的方法来快速交付，测试和部署代码，您可以大大减少编写代码和在生产环境中运行代码之间的延迟。

1、快速，一致地交付您的应用程序

Docker 允许开发人员使用您提供的应用程序或服务的本地容器在标准化环境中工作，从而简化了开发的生命周期。

容器非常适合持续集成和持续交付（CI / CD）工作流程，请考虑以下示例方案：

- 您的开发人员在本地编写代码，并使用 Docker 容器与同事共享他们的工作。
- 他们使用 Docker 将其应用程序推送到测试环境中，并执行自动或手动测试。
- 当开发人员发现错误时，他们可以在开发环境中对其进行修复，然后将其重新部署到测试环境中，以进行测试和验证。
- 测试完成后，将修补程序推送给生产环境，就像将更新的镜像推送到生产环境一样简单。

2、响应式部署和扩展

Docker 是基于容器的平台，允许高度可移植的工作负载。Docker 容器可以在开发人员的本机上，数据中心的物理或虚拟机上，云服务上或混合环境中运行。

Docker 的可移植性和轻量级的特性，还可以使您轻松地完成动态管理的工作负担，并根据业务需求指示，实时扩展或拆除应用程序和服务。

3、在同一硬件上运行更多工作负载

Docker 轻巧快速。它为基于虚拟机管理程序的虚拟机提供了可行、经济、高效的替代方案，因此您可以利用更多的计算能力来实现业务目标。Docker 非常适合于高密度环境以及中小型部署，而您可以用更少的资源做更多的事情。

相关链接

Docker 官网：https://www.docker.com
Github Docker 源码：https://github.com/docker/docker-ce

CentOS Docker 安装

Docker 支持以下的 64 位 CentOS 版本：

- CentOS 7
- CentOS 8
- 更高版本...

该 centos-extras 库必须启用。默认情况下，此仓库是启用的，但是如果已禁用它，则需要重新启用它。

建议使用 overlay2 存储驱动程序。

卸载旧版本

较旧的 Docker 版本称为 docker 或 docker-engine 。如果已安装这些程序，请卸载它们以及相关的依赖项。

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

安装 Docker Engine-Community

使用 Docker 仓库进行安装

在新主机上首次安装 Docker Engine-Community 之前，需要设置 Docker 仓库。之后，您可以从仓库安装和更新 Docker。

设置仓库

安装所需的软件包。yum-utils 提供了 yum-config-manager ，并且 device mapper 存储驱动程序需要 device-mapper-persistent-data 和 lvm2。

```
$ sudo yum install -y yum-utils \
  device-mapper-persistent-data \
  lvm2
```

使用以下命令来设置稳定的仓库。

```
sudo yum-config-manager \
    --add-repo \
    https://download.docker.com/linux/centos/docker-ce.repo
```

安装 Docker Engine-Community

安装最新版本的 Docker Engine-Community 和 containerd，或者转到下一步安装特定版本：

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

启动 Docker。

```
sudo systemctl start docker
```

镜像加速

Ubuntu16.04+、Debian8+、CentOS7

对于使用 systemd 的系统，请在 /etc/docker/daemon.json 中写入如下内容（如果文件不存在请新建该文件）：

```
{"registry-mirrors":["https://registry.docker-cn.com"]}
```

之后重新启动服务：

```
$ sudo systemctl daemon-reload
$ sudo systemctl restart docker
```

容器操作

1. 运行容器

使用命令：

```
docker run --name container-name:tag -d image-name
--name：自定义容器名，不指定时，docker 会自动生成一个名称
-d：表示后台运行容器
```

```
4 image-name：指定运行的镜像名称以及 Tag
```

2. 查看容器

使用命令：

```
1 docker ps 命令 查看正在运行的所有容器
2 加上 -a 参数可以查看所有容器（即无论是否运行中）
```

```
1 CONTAINER ID：容器 di
2 IMAGE：镜像名称:Tag
3  COMMAND：命令
4 CREATES：容器创建的时刻
5 STATUS：容器当前的状态（up 表示运行、Exited 表示停止运行）
6 PORTS：镜像程序使用的端口号
```

3. 停止容器

使用命令：

```
1 docker stop container-name/container-id
2 命令进行停止容器运行，指定容器名或者容器 id 即可
```

4. 启动容器

```
1 docker start container-name/container-id
2 命令启动停止运行的容器，同理可以根据 容器名或者 容器 id 进行启动
```

5. 删除容器

```
1 docker rm container-id 命令
2 删除容器，删除容器前，必须先停止容器运行，根据 容器 id 进行删除
3 rm 参数是删除容器，rmi 参数是删除镜像
4 镜像运行在容器中，docker 中可以运行多个互补干扰的容器，可以将同一个镜像在多个容器中进行运行
```

6. 端口映射

```
docker run --name container-name:tag -d -p 服务器端口:Docker端口 image-name
--name：自定义容器名，不指定时，docker 会自动生成一个名称
-d：表示后台运行容器
image-name：指定运行的镜像名称以及 Tag
-p 表示进行服务器与 Docker 容器的端口映射，默认情况下容器中镜像占用的端口是
 Docker 容器中的端口与外界是隔绝的，必须进行端口映射才能访问
启动成功之后，ip addr show 查一下服务器 ip 地址（ 192.168.58.129），然后就能从物理机上访问了
```

7. 容器日志

```
docker logs container-name/container-id 命令
可以查看容器日志信息，指定容器名或者 容器 id 即可
```

```
$ 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）
```

查看指定时间后的日志，只显示最后100行：

```
docker logs -f -t --since="2018-02-08" --tail=100 CONTAINER_ID
```

查看最近30分钟的日志:

```
docker logs --since 30m CONTAINER_ID
```

查看某时间之后的日志：

```
$ docker logs -t --since="2018-02-08T13:23:37" CONTAINER_ID
```

查看某时间段日志：

```
$ docker logs -t --since="2018-02-08T13:23:37" --until "2018-02-09T12:23:37" CONTAINER_ID
```

8. 复制内容

```
从主机复制到容器sudo docker cp host_path containerID:container_path
从容器复制到主机sudo docker cp containerID:container_path host_path
```

9. 数据卷

10.mysql

```
获取镜像
docker pull mysql:5.7
创建容器
docker run -d --name mysql-5.7-3306 -p 3306:3306 -e MESQL_ROOT_PASSWORD='root' images-id/images-name
进入容器
docker exec -it container_name bash
输入账号密码
mysql -u root -p
授权
grant all privileges on *.* to 'root'@'%' identified by 'root' with grant option;
刷新权限
flush privileges;
退出
exit;
```

11.CentOS防火墙端口

查看已经开放的端口：

```
firewall-cmd --list-port
```

开启端口：

```
firewall-cmd --zone=public --add-port=8080/tcp --permanent
```

重启防火墙：

```
firewall-cmd --reload #重启
firewall-cmd systemctl stop firewalld.service #停止
firewall-cmd systemctl disable firewalld.service #禁止firewall开机启动
```

12. Dockerfile 定制镜像

我们可以把刚才的对容器的所有操作命令都记录到一个文件里，就像写更脚本程序。之后用 docker build 命令以此文件为基础制作一个镜像，并会自动提交到本地仓库。这样的话镜像的构建会变的透明化，对镜像的维护起来也更加简单，只修改这个文件即可。同时分享也更加简单快捷，因为只要分享这个文件即可。Dokcerfile 是一个普通的文本文件，文件名一般叫 Dockerfile其中包含了一系列的指令(Instruction)， 每一条指令都会构建一层，就是描述该层是如何创建的。

12.1 dockerfile的常用指令

FROM	构造的新镜像是基于哪个镜像 例如：FROM centos:v1
MAINTAINER	维护者信息 例如：MAINTAINER yanglin
RUN	构建镜像时运行的shell命令 例如：RUN ["yum", "install", "http"] RUN yum install httpd
CMD	运行容器时执行的shell命令 例如：CMD ["-c","/startup.sh"] CMD ["/usr/sbin/sshd","-D"] CMD /usr/sbin/sshd -D
EXPOSE	指定于外界交互的端口，即容器在运行时监听的端口 EXPOSE 8081 8082
ENV	设置容器内环境变量 例如：ENV MYSQL_ROOT_PASSWORD 123456
COPY	拷贝文件或者目录到镜像，用法同ADD 例如：COPY ./startup.sh /startup.sh
ENTRYPOINT	运行容器时执行的shell命令 例如：ENTRYPOINT ["/bin/bash","-c","/startup.sh"] ENTRYPOINT /bin/bash -c '/startup.sh'
VOLUME	指定容器挂载点到宿主机自动生成的目录或者其他容器

	例如：VOLUME ["/path/to/dir"]
USER	为RUN,CMD,ENTRYPOINT执行命令指定运行用户 例如：USER www 镜像构建完成后，通过docker run运行容器时，可以通过-u参数来覆盖所指定的用户。
WORKDIR	为RUN,CMD,ENTRYPOINT,COPY和ADD设置工作目录 例如：WORKDIR /data
HEALTHCHECK	健康检查 HEALTHCHECK --interval=5m --timeout=3s CMD curl -f http://localhost/ ||exit 1
ADD	拷贝文件或者目录到镜像，如果是URL或者压缩包会自动下载或者自动解压 例如： ADD hom* /mydir/ ADD test relativeDir/
ARG	在构建镜像时指定一些参数 例如： FROM centos:6 ARG age=100

练习

1. 构建一个tomcat镜像

```
FROM tomcat:8.0
LABEL maintainer="sayonara_dh@163.com"
RUN echo 'hello daihang'>/usr/local/tomcat/webapps/ROOT/index.html
```

2. 在1的基础上删除ROOT里面的所有的文件，只保留index.html

   用到WORKDIR命令

```
FROM tomcat:8.0
LABEL maintainer="sayonara_dh@163.com"
WORKDIR /usr/local/tomcat/webapps/ROOT/
RUN rm -rf *
RUN echo 'hello daihang'>/usr/local/tomcat/webapps/ROOT/index.html
```

3. 在2的基础上，在webapps里面添加一张图片

```
FROM tomcat:8.0
LABEL maintainer="sayonara_dh@163.com"
WORKDIR /usr/local/tomcat/webapps/ROOT/
RUN rm -rf *
COPY docker.jpg /usr/local/tomcat/webapps/ROOT/
RUN echo 'hello daihang'>/usr/local/tomcat/webapps/ROOT/index.html
```

Docke部署Springboot项目

准备SpringBoot jar

Dockerfile

```
FROM java:8
LABEL maintainer="sayonara_dh@163.com"
VOLUME /tmp
ADD test01.jar test.jar
EXPOSE 8080
ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","test.jar"]
```

FROM：表示基础镜像，即运行环境

VOLUME /tmp创建/tmp目录并持久化到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

ADD：拷贝文件并且重命名(ADD exam-0.0.1-SNAPSHOT.jar exam.jar 将应用jar包复制到/exam.jar)

EXPOSE：并不是真正的发布端口，这个只是容器部署人员与建立image的人员之间的交流，即建立image的人员告诉容器布署人员容器应该映射哪个端口给外界

ENTRYPOINT：容器启动时运行的命令，相当于我们在命令行中输入java -jar xxxx.jar，为了缩短 Tomcat 的启动时间，添加java.security.egd的系统属性指向/dev/urandom作为 ENTRYPOINT

普通部署方式

1、在IDEA工具中开发代码。
2、代码打成jar包
3、部署到Linux服务器上
4、如果用Docker（ 编写Dockerfile文件）
5、构建镜像
6、运行容器

IDEA集成Docker实现镜像打包一键部署

1.开启docker远程访问（centOS）

- 首先编辑docker的宿主机文件/lib/systemd/system/docker.service

```
vi /lib/systemd/system/docker.service
```

- 修改以ExecStart开头的行：

```
ExecStart=/usr/bin/dockerd -H tcp://0.0.0.0:2375 -H unix:///var/run/docker.sock
```

- 如果是centos7以下的话，就把ExecStart修改为：

```
ExecStart=/usr/bin/dockerd -H fd:// -H tcp://0.0.0.0:2375
```

- 修改后保存文件，然后通知docker服务做出的修改

```
systemctl daemon-reload
```

- 重启docker服务

```
systemctl restart docker.servic
```

构建镜像

```
docker builde -t demo1 .
```