Docker

Docker 的基础是 Linux 容器（LXC）等技术。 Docker是一款针对程序开发人员和系统管理员来开发、部署、运行应用的一款虚拟化平台。 *Docker的运行需要Root权限，所以启动Docker的时候最好进行参数的安全检查*

简介

Docker 组件

- Docker Engine
- Docker Hub

Why is Docker?

- 开发人员不必关心开发环境的变化；
- Docker很轻量，启动迅速；
- 多环境支持，多硬件支持；

镜像

名词解释

- `基础镜像`：没有任何父镜像的镜像
- `镜像ID`: 所有镜像都是通过一个 64 位十六进制字符串 （内部是一个 256 bit 的值）来标识的。 为简化使用，前 12 个字符可以组成 一个短ID，可以在命令行中使用。短ID还是有一定的 碰撞机率，所以服务器总是返回长ID。

原理

```
Docker 镜像是怎么实现增量的修改和维护的？ 每个镜像都由很多层次构成，Docker 使用 Union FS 将这些不同的层结合到一个镜像中去。

通常 Union FS 有两个用途, 一方面可以实现不借助 LVM、RAID 将多个 disk 挂到同一个目录下,另一个更常用的就是将一个只读的分支和一个可写的分支联合在一起，Live CD 正是基于此方法可以允许在镜像不变的基础上允许用户在其上进行一些写操作。 Docker 在 AUFS 上构建的容器也是利用了类似的原理。
```

> AUFS是一种Union File System，所谓UnionFS就是把不同物理位置的目录合并mount到同一个目录中

> 典型的Linux文件系统由 `bootfs` 和 `rootfs` 两部分组成，bootfs(boot file system)主要包含 bootloader和kernel，bootloader主要是引导加载kernel， 当kernel被加载到内存中后 bootfs就被umount了。 rootfs (root file system) 包含的就是典型 Linux 系统中的/dev，/proc，/bin，/etc等标准目录和文件。

Docker文件系统层次结构

使用到的Linux底层技术

命名空间机制

> Linux 的命名空间机制提供了以下七种不同的命名空间，包括 `CLONE_NEWCGROUP` 、 `CLONE_NEWIPC` 、 `CLONE_NEWNET` 、 `CLONE_NEWNS` 、 `CLONE_NEWPID` 、 `CLONE_NEWUSER` 和 `CLONE_NEWUTS` ，通过这七个选项我们能在创建新的进程时设置新进程应该在哪些资源上与宿主机器进行隔离。

> 每一个使用 `docker run` 启动的容器其实都具有单独的网络命名空间，Docker 为我们提供了四种不同的网络模式，Host、Container、None 和 Bridge 模式。

> Docker 默认的网络设置模式：网桥模式。在这种模式下，除了分配隔离的网络命名空间之外，Docker 还会为所有的容器设置 IP 地址。当 Docker 服务器在主机上启动之后会创建新的虚拟网桥 docker0，随后在该主机上启动的全部服务在默认情况下都与该网桥相连。

命令

- 获取镜像：`docker pull ubuntu:12.04` 或 `docker pull registry.hub.docker.com/ubuntu:12.04`
- 运行镜像：`docker run -t -i ubuntu:12.04 /bin/bash`
  - *注：如果不指定具体的标记，则默认使用 `latest` 标记信息;*
  - `--name` 为容器命名
  - `-t` 选项让 `Docker` 分配一个伪终端 `pseudo-tty` ,并绑定到容器的标准输入上;
  - `-i` 则让容器的标准输入保持打开;
  - `-d` 以守护（ `deamon` ）进程的形式在后台运行docker，可以使用 `docker attach` 命令或 `nsenter` 工具进入守护容器；

```
$ sudo docker run -idt ubuntu
243c32535da7d142fb0e6df616a3c3ada0b8ab417937c853a9e1c251f499f550
$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS               NAMES
243c32535da7        ubuntu:latest       "/bin/bash"         18 seconds ago      Up 17 seconds
nostalgic_hypatia
$sudo docker attach nostalgic_hypatia
root@243c32535da7:/#
```

  - `-P` 或 `-p ip:hostPort:containerPort` 端口映射
    - `-p 127.0.0.1:5000:5000` 或 `5000:5000` 5000映射到5000
    - `-p 127.0.0.1::5000` 任意映射到5000
    - 可多次使用
- 查看当前映射的端口配置：`docker port`
- 列出本地镜像：`docker images`
- 提交更新后的副本：`docker commit -m "备注" -a "作者" 0b2616b0e5a8 ouruser/sinatra:v2`
  - `0b2616b0e5a8` 是用来创建镜像的容器的 ID
- 修改标签：`docker tag 5db5f8471261 ouruser/sinatra:devel`
- 从本地文件系统导入：传送门
- 上传镜像：`docker push ouruser/sinatra`
- 导出镜像到本地：`docker save -o ubuntu_14.04.tar ubuntu:14.04`
- 载入本地镜像：`docker load --input ubuntu_14.04.tar` 或 `docker load < ubuntu_14.04.tar`
- 从快照中导入镜像：`docker import` ，For Example：
  - `cat ubuntu.tar | sudo docker import - test/ubuntu:v1.0`
  - `docker import http://example.com/exampleimage.tgz example/imagerepo`
    注意：用户既可以使用 `docker load` 来导入镜像存储文件到本地镜像库，也可以使用 `docker import` 来导入一个容器快照到本地镜像库。这两者的区别在于容器快照文件将丢弃所有的历史记录和元数据信息（即仅保存容器当时的快照状

态），而镜像存储文件将保存完整记录，体积也要大。此外，从容器快照文件导入时可以重新指定标签等元数据信息。

- 移除容器：`docker rm`
- 移除镜像：`docker rmi training/sinatra`
  - 注意：*在删除镜像之前要先用* `docker rm` *删掉依赖于这个镜像的所有容器。*
- 启动运行已经终止的容器：`docker start`
- 查看容器运行信息：`docker ps`
  - `-a` 还能查看到已经终止的容器信息
- 获取容器输出信息：`docker logs`
- 终止容器：`docker stop`
- 重启运行中的容器：`docker restart`
- 查看所有变量：`docker inspect`

Dockerfile

语法

```
# This is a comment     # 注释
FROM ubuntu:14.04       # 基础镜像
MAINTAINER Docker Newbee <newbee@docker.com>    # 维护人
RUN apt-get -qq update  # 创建一层
RUN apt-get -qqy install ruby ruby-dev
RUN gem install sinatra
CMD /usr/sbin/nginx  # 指定运行容器时的操作指令
```

- 注意：*一个镜像不能超过 127 层*

数据卷

名词解释

- 数据卷是一个可供一个或多个容器使用的特殊目录，如果你有一些持续更新的数据需要在容器之间共享，最好创建数据卷容器
  - 可以在容器之间共享和重用
  - 修改会立马生效
  - 对数据卷的更新，不会影响镜像
  - 卷会一直存在，直到没有容器使用
- 数据卷容器：传送门

命令

- 创建数据卷：`docker run -d -P --name web -v /webapp[:/opt/webapp[:ro]] training/webapp python app.py`
  - `-v` 创建一个数据卷并挂载到容器里
  - `:/opt/webapp` 指定挂载到容器中的位置
  - `:ro` 只读（默认读写）

网络

- 容器互联：传送门
- 高级网络配置：传送门

安全介绍

- 安全配置值得注意：传送门