# Docker镜像

1. 我们知道镜像是Docker三大组件之一
2. Docker运行容器前需要本地存在对应的镜像
   1. 如果镜像不存在本地，Docker会从镜像仓库下载（默认是Docker Hub公共注册服务器中的仓库）

# 获取镜像

1. 可以使用**docker pull命令**从**仓库**获取所需要的镜像
   1. 比如sudo docker pull ubuntu:12.04，从Docker Hub仓库下载Ubuntu12.04镜像
   2. 下载过程中，会输出获取镜像的每一层信息
   3. 该命令实际相当于sudo docker pull registry.hub.docker.com/ubuntu:12.04命令，即从注册服务器registry.hub.docker.com中的ubuntu仓库下载标记为12.04的镜像
2. 有时候官方仓库注册服务器下载较慢，可以从其他仓库下载。从其他仓库下载时需要指定完整的仓库注册服务器地址
   1. 比如sudo docker pull dl.dockerpool.com:5000/ubuntu:12.04
3. 镜像获取完成后，即可随时使用该镜像了
   1. 例如创建一个容器，让其中运行bash应用：
      1. sudo docker run -t -i ubuntu:12.04 /bin/bash

# 列出本地的镜像

1. 使用**docker images命令**显示本地已有的镜像
2. 可以看到有几个字段信息：
   1. REPOSITORY，来自哪个仓库，相当于下载的是什么软件
   2. TAG，镜像的标记，相当于哪个版本
   3. IMAGE ID，镜像的ID，唯一标识。如果镜像ID相同，则是同一镜像
   4. CREATED，创建时间
   5. VIRTRUAL SIZE，镜像大小
3. 启动镜像时，如果不指定具体标记，就默认使用latest标记信息，即最新版本

# 创建镜像

1. 创建镜像有很多方法，用户可以从Docker Hub获取已有镜像并更新，也可以利用本地文件系统创建一个

**修改已有镜像**

1. 先使用下载的镜像启动容器
   1. 比如:sudo docker run -t -i training/sinatra /bin/bash
   2. 命令行标识符变为：root@0b2616b0e5a8:/#
   3. 记住容器的id，稍后还会用到
2. 在容器添加json和gem两个应用：
   1. root@0b2616b0e5a8:/# gem install json
3. 使用**exit命令**退出容器
4. 使用**docker commit命令**提交更新后的副本
   1. sudo docker commit -m “information” -a “info” id reposityName:tag
   2. -m用来指定提交的说明信息
   3. -a指定更新的用户信息，创建镜像的容器的ID，目标镜像的仓库名和tag信息

**利用Dockerfile来创建镜像**

1. 使用docker commit来扩展一个对象比较简单，但不方便在一个团队中分享。可以使用docker build创建一个新的镜像。因此，需要首先创建一个新的Dockerfile
2. 新建一个目录sinatra和一个Dockerfile
   1. mkdir sinatra
   2. cd sinatra
   3. touch Dockerfile
3. Dockerfile中每一条指令都创建镜像的一层
4. Dockerfile的基本语法是：
   1. 使用#来注释
   2. FROM指令告诉Docker**使用哪个镜像作为基础**
   3. 然后是维护者的信息：MAINTAINER Docker name <email>
   4. RUN开头的指令会在创建中运行，比如安装一个软件包
5. 编写完成Dockerfile后使用**docker build命令**生成镜像
   1. 例如sudo docker build -t=”.:tag”
   2. -t标记用来添加tag，指定新的镜像的用户信息，”.”是Dockerfile所在路径（当前目录），也可以替换为一个具体的Dockerfile的路径
   3. 可以看到build进程做的第一件事是上传Dockerfile内容
   4. 然后Dockerfile中的指令被一条条执行。每一步都创建了一个新的容器，在容器中执行指令并提交修改。
   5. 当所有指令都执行完毕，返回最终的镜像id。删除和清理所有中间步骤产生的容器
6. \*注，一个镜像不能超过127层
7. 在Dockerfile中，还可以：
   1. 利用ADD命令复制本地文件到镜像，比如ADD myAPP /var/image
   2. 利用EXPOSE命令向外部开放端口，比如EXPOSE 80
   3. 用CMD命令描述容器启动后运行的程序，CMD [“/bin/bash”, “-D”, “FOREGROUND]
8. 可以用**docker tag命令**修改镜像的标签
   1. 比如，sudo docker tag <imageId> <repositoryName>:<tag>

**从本地文件系统导入**

1. 可以使用openvz（容器虚拟化的先锋技术）的模板来创建
2. 比如先下载了一个ubuntu-14.04的镜像，然后使用**docker import命令**导入：
   1. sudo cat ubuntu-14.04-x86\_64.tar.gz | docker import - ubuntu:14.04

# 上传镜像

1. 用户可以通过docker push命令，把自己创建的镜像上传到仓库中来共享。
2. 比如，用户在Docker Hub上完成注册后，可以推送自己的镜像example/image到仓库中：
   1. sudo docker push example/image

# 导出镜像到本地文件

1. 使用**docker save命令**可以将镜像导出到本地文件
2. 比如，docker save -o ubuntu\_14.04.tar ubuntu:14.04，将镜像ubuntu:14.04保存到了本地文件ubuntu\_14.04.tar中

# 从本地文件载入本地镜像库

1. 可以使用docker load命令从镜像导出的本地文件中再导入到本地镜像库
   1. 比如，sudo docker load --input ubuntu\_14.04.tar
   2. 或者，sudo docker load < ubuntu\_14.04.tar
2. 将导入镜像以及相关的元数据信息（包括标签等）

# 移出本地镜像

1. 使用**docker rmi命令**可以移出本地镜像
   1. 比如，sudo docker rmi training/sinatra，移除了training/sinatra这个镜像
2. 使用**docker rm命令**可以移出容器
3. **\*注意，在移出镜像之前要先用docker rm删除掉依赖于这个镜像的所有容器**

# **镜像的实现原理**

1. 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
2. 通常Union FS有两个用途
   1. 实现不借助LVM、RAIN将多个disk挂到同一个目录下
   2. 将一个只读的分支和一个可写的分支联合在一起，Live CD正式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
3. Docker在AUFS上够贱的容器也是利用了类似的原理