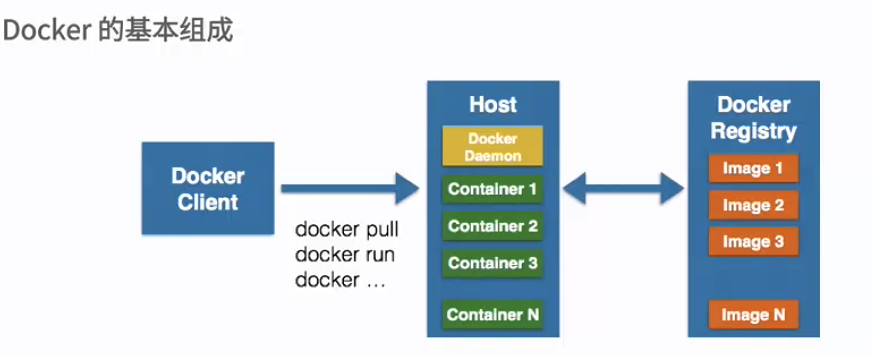
## Docker学习笔记

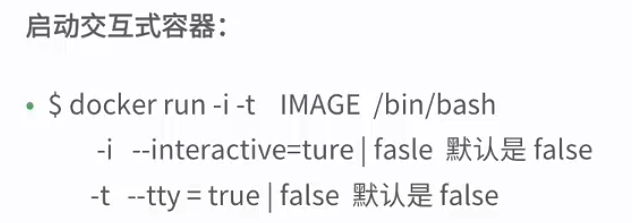
Docker:容器技术

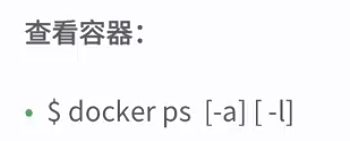
**Docker run xxx**

**在run后就生成一个新的容器**









-a是查看所有容器，-l是查看最新的容器



命令1：docker search xxx 在docker仓库中搜索容器镜像



命令2：docker pull xxx 在docker仓库中pull容器镜像



命令3：docker run xxx + 在容器中执行指令



命令4：docker commit 容器ID 新镜像名

命令5：docker inspect 容器ID/NAME 查看该容器详细信息

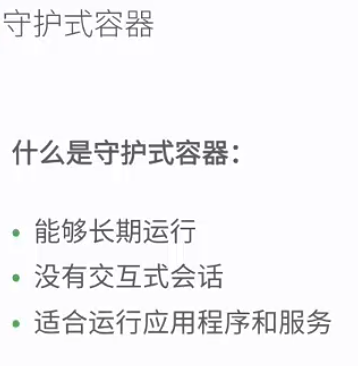


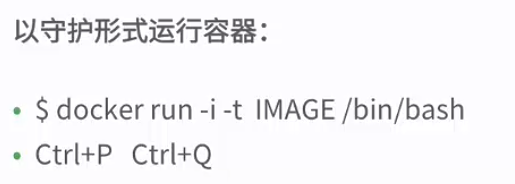
命令6：run - -name=xxx [OPTS] IMAGE [CMD] [ARGS...] 给新的容器自定义命名

在运行一个命令时可以没必要建立一个新的容器

命令7：重新启动一个已经停止了的容器: docker start [-i] [以交互的方式]

命令8：删除一个容器： docker rm 容器ID/NAME





这里有一个偷懒的写法：docker run -it IMAGE /bin/bash (即用交互命令行方式运行容器)

即容器在后台运行，当将后台转前台时，使用docker attach ID/NAME

（可以先通过docker ps -a 来查看所有的容器）

启动守护式容器： docker run -d IMAGE [CMD] [ARGS...]



对于后台运行的容器，将其转为前台可交互命令行（参见**在运行中的容器内启动新的进程**）：docker exec -it ID/NAME /bin/bash



**查看容器日志**

命令：docker logs [-f/--follows] [-t/--timestampes] [--tail] ID/NAME

其中：-f=true|false 是否一直跟踪日志的变化并返回结果 默认false

-t=true|false 是否在返回的结果上返回时间戳 默认false

--tail=‘all’ 返回结尾处多少数量的日志 默认all

**查看运行容器中的进程情况**

命令：docker top ID/NAME

**在运行中的容器内启动新的进程**

命令：docker exec [-d] [-i] [-t] NAME/ID [CMD] [ARGS...]

启动交互式bash，也是在容器中启动一个进程

**停止容器**

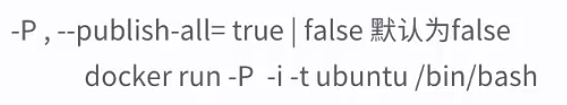
命令：docker stop ID/NAME 缓慢停止

Docker kill ID/NAME 强制停止

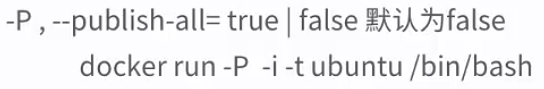
**在容器中部署静态网站**

1. **设置容器的端口映射**

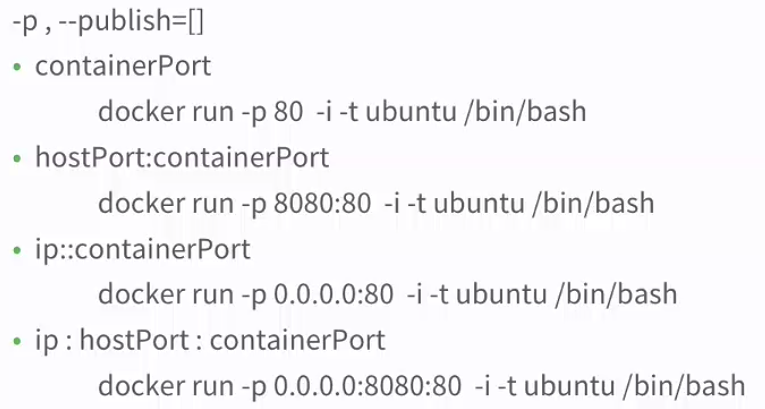
命令：run [-P] [-p]



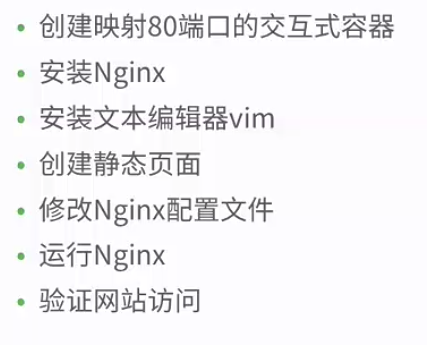
-P为docker自动分配端口（暴露可用的端口）



-p 手动分配端口，四种模式



1. **Nginx部署流程**



1. 创建

命令：docker run -p 80 --name web -it ubuntu /bin/bash

1. 安装nginx

命令：apt-get install -y nginx (-y表示安装， 若遇到错误，则先apt-get update更新源一下)

1. 安装vim编辑器 apt-get install -y vim
2. 创建静态页面
   1. 创建多级目录/var/www/html 命令：mkdir -p /var/www/html

...

1. 修改nginx配置文件

命令;先查找nginx位置，通过whereis nginx 查找包含nginx的文件夹

在ubuntu中 vi /etc/nginx/sites-enable/default 修改即可 然后nginx启动 用ps -ef查看所有进程

1. 查看映射

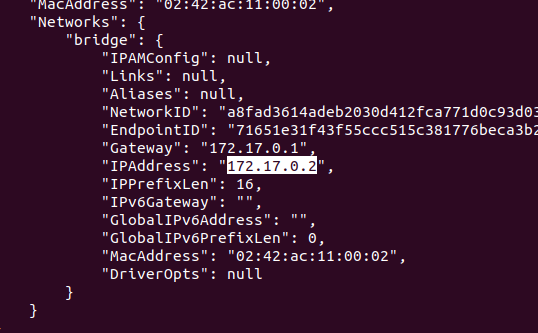
命令：docker port ID/NAME 或者 docker ps 或者 docker top ID/NAME



将容器中的web80端口映射到宿主机的32770端口

检测状态可以用浏览器访问 <http://localhost:32770> 或者命令行指令 cur <http://localhost:32770>

也可以通过docker容器分配的ip来访问 通过docker inspect ID/NAME 来查看IPAddress来查看该container分配的IP

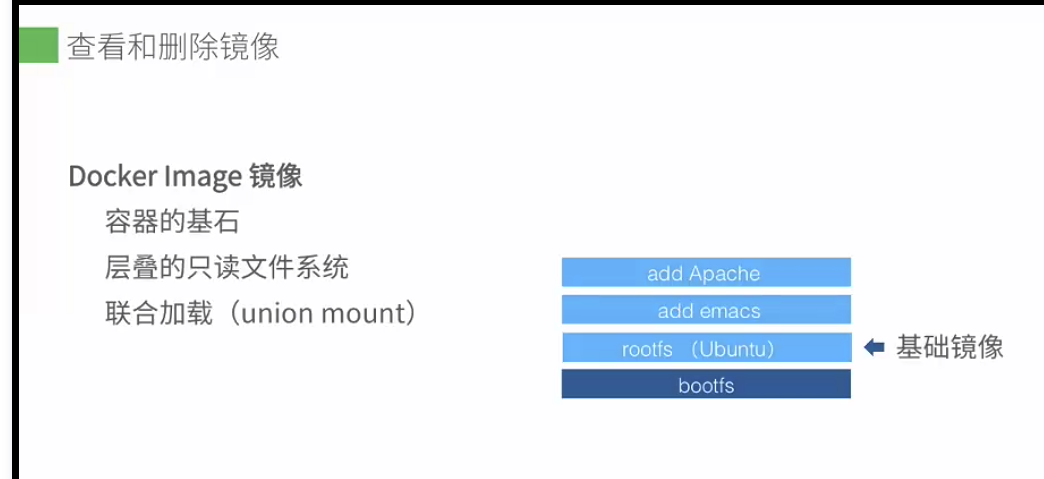


也可以通过容器的ip访问： curl <http://172.17.0.2> (因为默认80端口)

注意：

在通过docker stop/kill NAME/ID 后，通过docker start [OPTS] ID/NAME 后

还需启动nginx 如：docker exec ID/NAME nginx



运用docker info命令查看docker的存储驱动



-a：是否列出所有镜像

-f：在显示时的过滤条件

--no-trunc：是否不截断IMAGE的ID

-q：只显示IMAGE的唯一ID



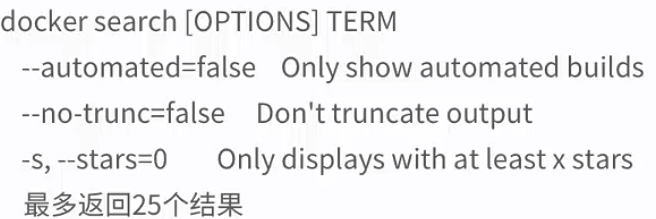


删除所有IMAGE

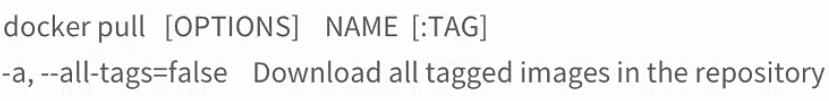


**获取和推送IMAGE（本地仓库和远程REGISTRY仓库的互动）**

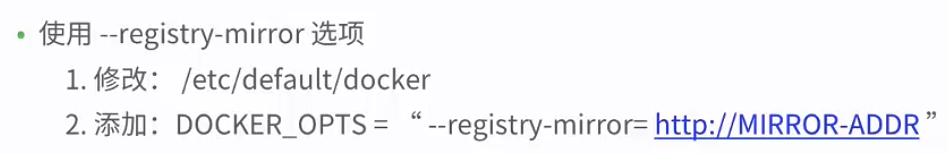
1. **查找IMAGE**



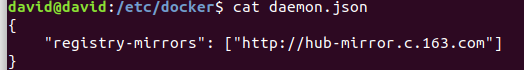
1. **拉取IMAGE**



1. 配置远端REGISTRY镜像源



**或者**在目录/etc/docker/ 下新建daemon.json

配置：

配置好后重新启动docker： service docker restart

1. 把自己配置好的容器提交为本地的IMAGE

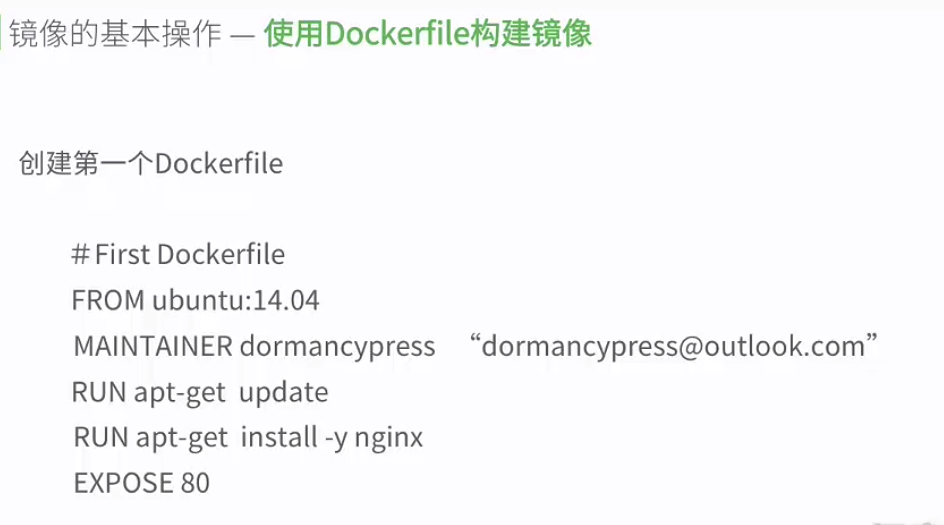
参见命令4：docker commit 容器ID 新镜像名

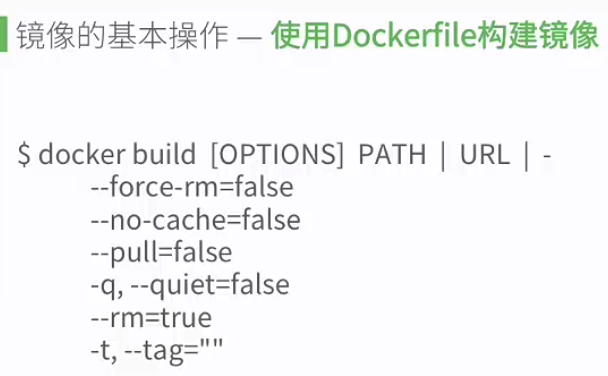
1. 将本地的IMAGE上传至dockerhub

Docker push IMAGE

**构建IMAGE**







-t：是给IMAGE取一个名字

**Docker的远程访问**

**（注意保持远程docker版本与本地docker版本一致）**

**安装docker后本地会有cli和server端，默认均为unix的socket连接。**

**要使得本地的docker的cli连接远程的docker的daemon SERVER。**

1. **设置远程docker服务器为tcp连接**

**（ubuntu环境，在/etc/default/docker文件下设置DOCKER\_OPTS=“ --label name=docker\_remote\_server -H tcp://0.0.0.0:2375” ubuntu 较高版本的docker配置 为修改/lib/systemd/system/docker.service 下的ExecStart项）**

**改为unix的socket连接就改成fd://**

1. **修改客户端的连接方式**

**使用环境变量： export DOCKER\_HOST=”tcp://x.x.x.x:2375”**

**改为本地只需要 export DOCKER\_HOST= 即可**