Docker 包括三个基本概念

* 镜像（Image）
* 容器（Container）
* 仓库（Repository）

镜像（Image）和容器（Container）的关系，就像是面向对象程序设计中的 类 和 实例 一样，镜像是静态的定义，容器是镜像运行时的实体。容器可以被创建、启动、停止、删除、暂停等。

Dockerfile命令：

docker build -t nginx:v3 .

Docker build: -f参数指定dockerfile文件

.代表上下文目录，也是当前目录，类似但并非dockerfile的目录。

COPY：

COPY hom\* /mydir/

COPY hom?.txt /mydir/

当利用 docker run 来创建容器时，Docker 在后台运行的标准操作包括：

* 检查本地是否存在指定的镜像，不存在就从公有仓库下载
* 利用镜像创建并启动一个容器
* 分配一个文件系统，并在只读的镜像层外面挂载一层可读写层
* 从宿主主机配置的网桥接口中桥接一个虚拟接口到容器中去
* 从地址池配置一个 ip 地址给容器
* 执行用户指定的应用程序
* 执行完毕后容器被终止

Docker-compose与dockerfile区别：

我所理解的docker-compose是编排容器的。例如，你有一个php镜像，一个mysql镜像，一个nginx镜像。如果没有docker-compose，那么每次启动的时候，你需要敲各个容器的启动参数，环境变量，容器命名，指定不同容器的链接参数等等一系列的操作，相当繁琐。而用了docker-composer之后，你就可以把这些命令一次性写在docker-composer.yml文件中，以后每次启动这一整个环境（含3个容器）的时候，你只要敲一个docker-composer up命令就ok了。

而dockerfile的作用是从无到有的构建镜像。

Docker-compose编写

运行：

docker-compose build

#### docker-compose up -d

#### Docker单独命令映射到管理命令

1.12    1.13                用途

attach  container attach    附加到一个运行的容器

build   image build         从一个Dockerfile构建镜像

commit  container commit    从一个容器的修改创建一个新的镜像

cp      container cp        在容器与本地文件系统之间复制文件/文件夹

create  container create    创建新的容器

diff    container diff      检阅一个容器文件系统的修改

events  system events       获取服务器的实时时间

exec    container exec      在运行的容器内执行命令

export  container export    打包一个容器文件系统到tar文件

history image history       展示镜像历史信息

images  image ls            展示镜像列表

import  image import        用tar文件导入并创建镜像文件

info    system info         展示整个系统信息

inspect container inspect   展示一个容器/镜像或者任务的底层信息

kill    container kill      终止一个或者多个运行中的容器

load    image load          从tar文件或者标准输入载入镜像

login   login               登录Docker registry

logout  logout              从Docker registry登出

logs    container logs      获取容器的日志

network network             管理Docker网络

node    node                管理Docker Swarm节点

pause   container pause     暂停一个或者多个容器的所有进程

port    container port      展示容器的端口映射

ps      container ls        展示容器列表

pull    image pull          从某个registry拉取镜像或者仓库

push    image push          推送镜像或者仓库到某个registry

rename  container rename    重命名容器

restart container restart   重启容器

rm      container rm        移除一个或多个容器

rmi     image rm            移除一个或多个镜像

run     container run       运行一个新的容器

save    image save          打包一个或多个镜像到tar文件(默认是到标准输出)

search  search              在Docker Hub搜索镜像

service service             管理Docker services

start   container start     启动一个或者多个容器

stats   container stats     获取容器的实时资源使用统计

stop    container stop      停止一个或多个运行容器

swarm   swarm               管理Docker Swarm

tag     image tag           标记一个镜像到仓库

top     container top       展示容器运行进程

unpause container unpause   解除暂停一个或多个容器的所有进程

update  container update    更新一个或多个容器的配置

version version             显示Docker版本信息

volume  volume              管理Docker volumes

wait    container wait      阻塞直到容器停止，然后打印退出代码

部署题目，pull镜像直接给外网借口：

docker pull xx/xx / docker build -t xx/xx .

docker run -p 80:80 -v /data:/data -d xx/xx

杀死运行中的容器:

docker kill -s KILL ContainerId

删除容器(-f杀死运行中的):  
docker rm id

删除镜像:  
docker rmi id

批量:

删除所有容器 docker rm $(docker ps -a -q)  
  
删除所有镜像 docker rmi $(docker images -q)

查看容器日志: docker logs -f -t --tail 行数 容器名