Docker

1、docker容器的优势

1. 镜像、容器和仓库

Docker 由镜像(Image)、容器(Container)、仓库(Repository) 三部分组成。

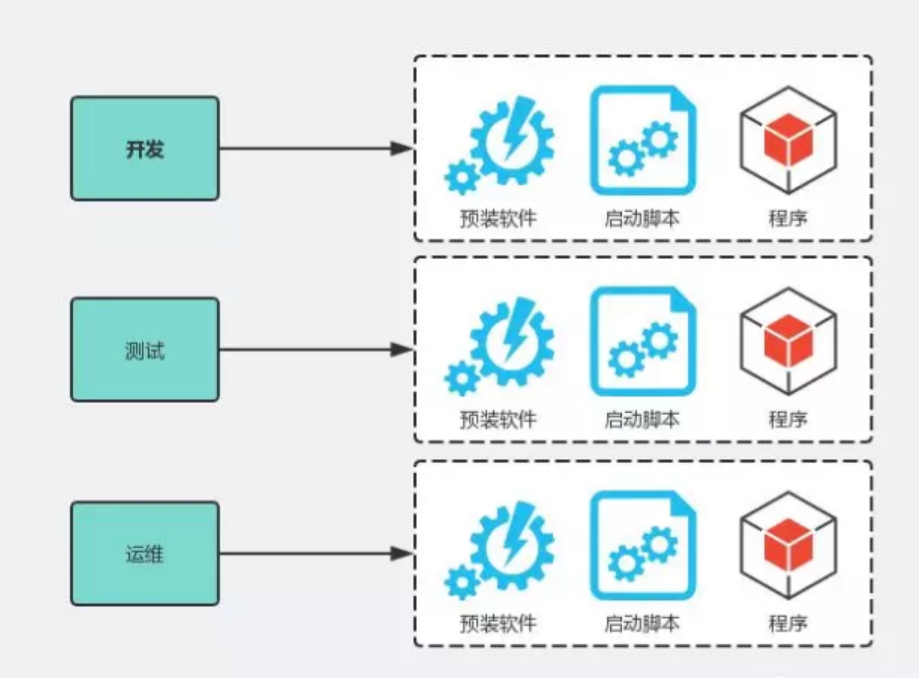
Docker 的镜像可以简单的类比为电脑装系统用的系统盘，包括操作系统，以及必要的软件。例如，一个镜像可以包含一个完整的 centos 操作系统环境，并安装了 Nginx 和 Tomcat 服务器。注意的是，镜像是只读的。这一点也很好理解，就像我们刻录的系统盘其实也是可读的。我们可以使用 docker images 来查看本地镜像列表。

Docker 的容器可以简单理解为提供了系统硬件环境，它是真正跑项目程序、消耗机器资源、提供服务的东西。例如，我们可以暂时把容器看作一个 Linux 的电脑，它可以直接运行。那么，容器是基于镜像启动的，并且每个容器都是相互隔离的。注意的是，容器在启动的时候基于镜像创建一层可写层作为最上层。我们可以使用 docker ps-a 查看本地运行过的容器。

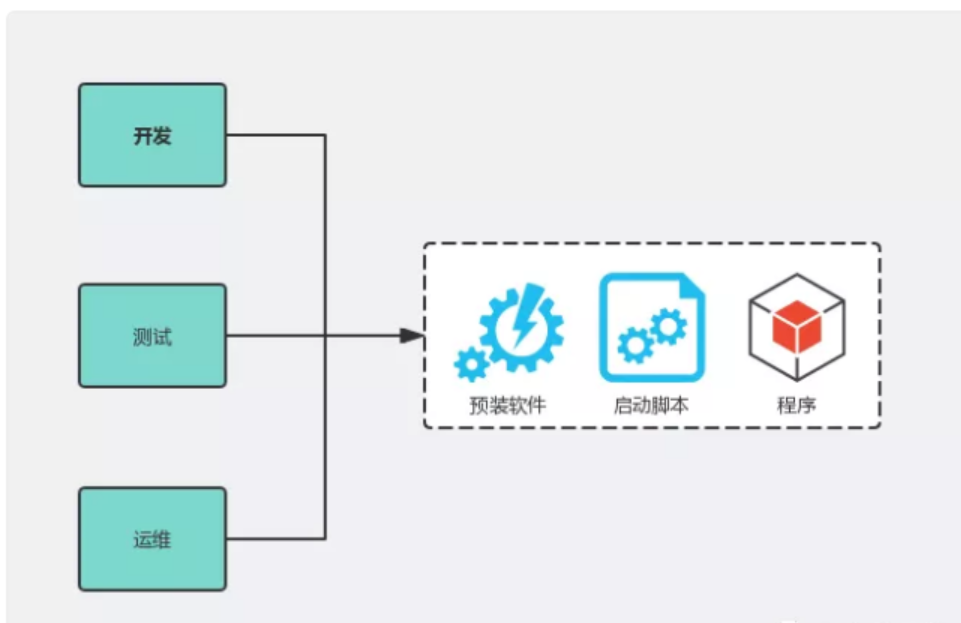
Docker 的仓库用于存放镜像。这一点，和 Git 非常类似。我们可以从中心仓库下载镜像，也可以从自建仓库下载。同时，我们可以把制作好的镜像 commit 到本地，然后 push 到远程仓库。仓库分为公开仓库和私有仓库，最大的公开仓库是官方仓库 Dock Hub，国内的公开仓库也有很多选择，例如阿里云等。

### 3. Docker 促使开发流程变更

Docker 对开发流程的影响在于使环境标准化。例如，原来我们存在三个环境：开发（日常）环境、测试环境、生产环境。这里，我们对于每个环境都需要部署相同的软件、脚本和运行程序，如图所示。事实上，对于启动脚本内容都是一致的，但是没有统一维护，经常会出问题。此外，对于运行程序而言，如果所依赖的底层运行环境不一致，也会造成困扰和异常。



现在，我们通过引入 Docker 之后，我们只需要维护一个 Docker 镜像。换句话说，多套环境，一个镜像，实现系统级别的一次构建到处运行。此时，我们把运行脚本标准化了，把底层软件镜像化了，然后对于相同的将要部署的程序实行标准化部署。因此，Docker 为我们提供了一个标准化的运维模式，并固化运维步骤和流程。



通过这个流程的改进，我们更容易实现 DevOps 的目标，因为我们的镜像生成后可以跑在任何系统，并快速部署。此外，使用 Docker 的很大动力是基于 Docker 实现弹性调度，以更充分地利用机器资源，节省成本。

在使用 Docker 过程中，还发现了一些很棒的收益点，例如我们发布回滚的时候只需要切换 TAG 并重启即可。还比如，我们对环境升级，也只需要升级基础镜像，那么新构建的应用镜像，自动会引用新的版本。

1. 搭建 Web 服务器

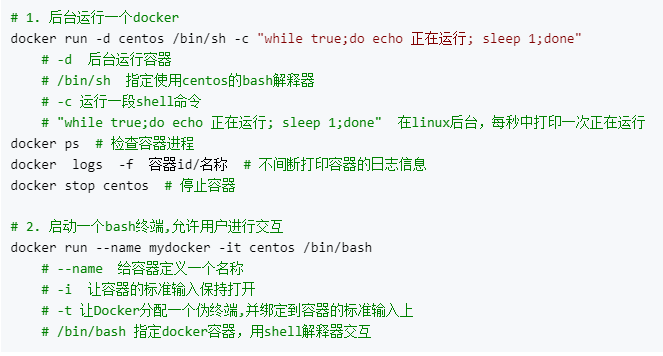
### 安装 Docker



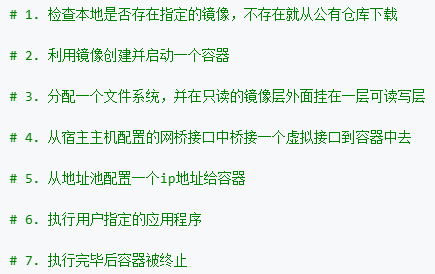
1. **设置加速服务（重启docker）**

curl -sSL https://get.daocloud.io/daotools/set\_mirror.sh | sh -s http://95822026.m.daocloud.io

1. **启动容器**
2. 基于镜像新建一个容器并启动



当利用docker run来创建容器时，Docker在后台运行的步骤如下：



1. **搭建web服务**
2. 安装 centos 镜像



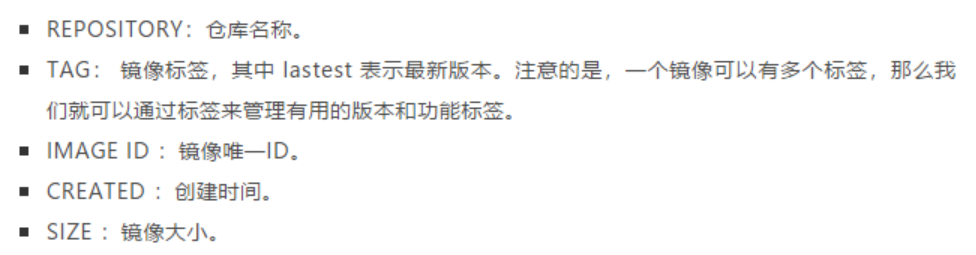
-p 80 Docker 开放 80 端口，宿主机器会自动做端口映射，如果关闭或者重启，端口会修改。

docker run 。Docker 通过 run 命令来启动一个新容器。Docker 首先在本机中寻找该镜像，如果没有安装，Docker 在 Docker Hub 上查找该镜像并下载安装到本机，最后 Docker 创建一个新的容器并启动该程序。第二次执行  docker run 时， Docker 会直接创建一个新的容器并启动该程序。docker run 每次使用都会创建一个新的容器，因此，我们以后再次启动这个容器时，只需要使用命令 docker start  即可docker start 的作用在用重新启动已存在的镜像，而 docker run 包含将镜像放入容器中 docker create ，然后将容器启动 docker start

docker exit -it 容器id /bin/bash  关闭 Docker 容器，进入容器交互式界面（后台的守护进程，通过 docker stop web ）

docker attach 切换到运行交互式容器





docker pull centos:latest 拉取镜像

 搜索NGINX

1. 安装NGINX



输入 whereis nginx 命令就可以看到安装的路径，启动NGINX



执行 ctrl+P+Q 切换到后台。然后，通过 docker ps-a 来查看随机分配的端口。

## 构建我的镜像





FROM centos:7：从一个已经存在的镜像运行一个容器

MAINTAINER ： 指定该镜像的作者

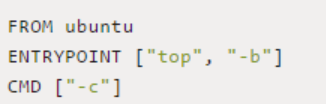
EXPOSE80 暴露 Dokcer 容器的 80 端口



docker build 命令进行构建，指定 -t 告诉 Docker 镜像的名称和版本。注意的是，如果没有指定任何标签，Docker 将会自动为镜像设置一个 lastest 标签。 . 是为了让 Docker 到当前本地目录去寻找 Dockerfile 文件。注意的是，Docker 会在每一步构建都会将结果提交为镜像，然后将之前的镜像层看作缓存，因此我们重新构建类似的镜像层时会直接复用之前的镜像。如果我们需要跳过，可以使用 --no-cache 选项告诉 Docker 不进行缓存。

指令辨别一：RUN、CMD、ENTRYPOINT

RUN 、 CMD 、 ENTRYPOINT  三个指令的用途非常相识，不同在于， RUN 指令是在容器被构建时运行的命令，而 CMD 、 ENTRYPOINT 是启动容器时执行 shell 命令，而 RUN 会被 docker run 命令覆盖，但是  ENTRYPOINT 不会被覆盖。事实上， docker run 命令指定的任何参数都会被当作参数再次传递给 ENTRYPOINT  指令。 CMD 、 ENTRYPOINT 两个指令之间也可以一起使用。例如，我们 可以使用 ENTRYPOINT 的 exec 形式设置固定的默认命令和参数，然后使用任一形式的 CMD 来设置可能更改的其他默认值。



指令辨别二：ADD、COPY

ADD 、 COPY 指令用法一样，唯一不同的是 ADD  支持将归档文件（tar, gzip, bzip2, etc）做提取和解压操作。注意的是， COPY 指令需要复制的目录一定要放在 Dockerfile 文件的同级目录下。

1. 将镜像推送到远程仓库

远程仓库：阿里云

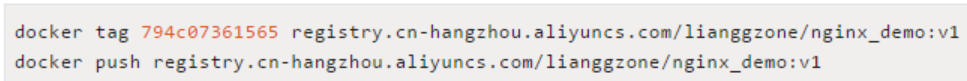
首先，在终端中输入访问凭证，登录 Registry 实例。

<https://cr.console.aliyun.com/cn-hangzhou/instances/credentials。>



将镜像推送到阿里云镜像仓库。其中，

docker tag[IMAGE\_ID]registry.cn-hangzhou.aliyuncs.com/[命名空间]/[镜像名称]:[版本] 和 docker push registry.cn-hangzhou.aliyuncs.com/[命名空间]/[镜像名称]:[版本] 使用如下：



上传完成后访问：

<https://cr.console.aliyun.com/cn-hangzhou/instances/repositories>

## Docker基础命令

Options:

--config=~/.docker Location of client config files #客户端配置文件的位置

-D, --debug=false Enable debug mode #启用Debug调试模式

-H, --host=[] Daemon socket(s) to connect to #守护进程的套接字（Socket）连接

-h, --help=false Print usage #打印使用

-l, --log-level=info Set the logging level #设置日志级别

--tls=false Use TLS; implied by--tlsverify #

--tlscacert=~/.docker/ca.pem Trust certs signed only by this CA #信任证书签名CA

--tlscert=~/.docker/cert.pem Path to TLS certificate file #TLS证书文件路径

--tlskey=~/.docker/key.pem Path to TLS key file #TLS密钥文件路径

--tlsverify=false Use TLS and verify the remote #使用TLS验证远程

-v, --version=false Print version information and quit #打印版本信息并退出

Commands:

attach Attach to a running container #当前shell下attach连接指定运行镜像

build Build an image from a Dockerfile #通过Dockerfile定制镜像

commit Create a new image from a container's changes #提交当前容器为新的镜像

cp Copy files/folders from a container to a HOSTDIR or to STDOUT #从容器中拷贝指定文件或者目录到宿主机中

create Create a new container #创建一个新的容器，同run 但不启动容器

diff Inspect changes on a container's filesystem #查看docker容器变化

events Get real time events from the server#从docker服务获取容器实时事件

exec Run a command in a running container#在已存在的容器上运行命令

export Export a container's filesystem as a tar archive #导出容器的内容流作为一个tar归档文件(对应import)

history Show the history of an image #展示一个镜像形成历史

images List images #列出系统当前镜像

import Import the contents from a tarball to create a filesystem image #从tar包中的内容创建一个新的文件系统映像(对应export)

info Display system-wide information #显示系统相关信息

inspect Return low-level information on a container or image #查看容器详细信息

kill Kill a running container #kill指定docker容器

load Load an image from a tar archive or STDIN #从一个tar包中加载一个镜像(对应save)

login Register or log in to a Docker registry#注册或者登陆一个docker源服务器

logout Log out from a Docker registry #从当前Docker registry退出

logs Fetch the logs of a container #输出当前容器日志信息

pause Pause all processes within a container#暂停容器

port List port mappings or a specific mapping for the CONTAINER #查看映射端口对应的容器内部源端口

ps List containers #列出容器列表

pull Pull an image or a repository from a registry #从docker镜像源服务器拉取指定镜像或者库镜像

push Push an image or a repository to a registry #推送指定镜像或者库镜像至docker源服务器

rename Rename a container #重命名容器

restart Restart a running container #重启运行的容器

rm Remove one or more containers #移除一个或者多个容器

rmi Remove one or more images #移除一个或多个镜像(无容器使用该镜像才可以删除，否则需要删除相关容器才可以继续或者-f强制删除)

run Run a command in a new container #创建一个新的容器并运行一个命令

save Save an image(s) to a tar archive#保存一个镜像为一个tar包(对应load)

search Search the Docker Hub for images #在dockerhub中搜索镜像

start Start one or more stopped containers#启动容器

stats Display a live stream of container(s) resource usage statistics #统计容器使用资源

stop Stop a running container #停止容器

tag Tag an image into a repository #给源中镜像打标签

top Display the running processes of a container #查看容器中运行的进程信息

unpause Unpause all processes within a container #取消暂停容器

version Show the Docker version information#查看容器版本号

wait Block until a container stops, then print its exit code #截取容器停止时的退出状态值

Run 'docker COMMAND --help' for more information on a command. #运行docker命令在帮助可以获取更多信息

docker search hello-docker # 搜索hello-docker的镜像

docker search centos # 搜索centos镜像

docker pull hello-docker # 获取centos镜像

docker run hello-world #运行一个docker镜像，产生一个容器实例（也可以通过镜像id前三位运行）

docker image ls # 查看本地所有镜像

docker images # 查看docker镜像

docker image rmi hello-docker # 删除centos镜像

docker ps #列出正在运行的容器（如果创建容器中没有进程正在运行，容器就会立即停止）

docker ps -a # 列出所有运行过的容器记录

docker save centos > /opt/centos.tar.gz # 导出docker镜像至本地

docker load < /opt/centos.tar.gz #导入本地镜像到docker镜像库

docker stop `docker ps -aq` # 停止所有正在运行的容器

docker rm `docker ps -aq` # 一次性删除所有容器记录

docker rmi `docker images -aq` # 一次性删除所有本地的镜像记录