# 一、搭建私有仓库

## 安装搭建

1. 需要安装docker 略
2. 使用docker registry镜像来搭建私有仓库，可以在github下载源码（<https://github.com/docker/docker-registry>）使用docker build创建镜像（详见[DockerFile方式](#_DockerFile方式)），也可以直接使用dockerHub上镜像docker pull registry，
3. 启动镜像 docker run –p 5000：5000 register

## 学习记录

1. 该镜像的仓库支持多种存储方式：具体详见github上的readme

主要是通过docker run –e参数设置环境变量来运行不同的存储方式

这些环境变量可以参考config文件夹的config\_sample.yml文件

其他参数-p –v等都需要学习一下，实际运行命令：docker run -d -e SETTINGS\_FLAVOR=local -e STORAGE\_PATH=/mnt/registry -e CACHE\_REDIS\_HOST=10.158.240.100 -e CACHE\_REDIS\_PORT=6379 -p 5000:5000 -v /mnt/registry:/mnt/registry registry:2

oss私有镜像存储方案参考：<https://help.aliyun.com/knowledge_detail/5974867.html?spm=5176.7114037.1996646101.5.umWUdZ&pos=3>

1. 由于国内dockerHub被墙，所以使用daocloud的docker镜像

设置方法见https://dashboard.daocloud.io/mirror

1. 这个registy内部有redis里面提供了redis和redis lru缓存可配置，对于并发请求的处理应该能够快速处理，有待确定
2. docker使用http方式上传镜像，将10.158.240.100：5000这个镜像点设置为信任镜像点，使用http方式，设置方法：

centos vi /etc/sysconfig/docker

修改为other\_args="--insecure-registry 10.158.240.100:5000"

ubuntu vi /etc/default/docker

DOCKER\_OPTS="--insecure-registry 10.158.240.100:5000"

1. docker inspect 查看docker容器变量

# 二、制作docker镜像

### DockerFile方式

Dockerfile用来创建一个自定义的image,包含了用户指定的软件依赖等。因此掌握Dockerfile的语法就可以创建一个镜像，官方参考: <https://docs.docker.com/engine/articles/dockerfile_best-practices/>

#### 学习记录

命令使用

Usage: docker build [OPTIONS] PATH | URL | -

Build a new image from the source code at PATH

-c, --cpu-shares=0 CPU shares (relative weight)

--cgroup-parent= Optional parent cgroup for the container

--cpu-period=0 Limit the CPU CFS (Completely Fair Scheduler) period

--cpu-quota=0 Limit the CPU CFS (Completely Fair Scheduler) quota

--cpuset-cpus= CPUs in which to allow execution (0-3, 0,1)

--cpuset-mems= MEMs in which to allow execution (0-3, 0,1)

--disable-content-trust=true Skip image verification

-f, --file= Name of the Dockerfile (Default is 'PATH/Dockerfile')

--force-rm=false Always remove intermediate containers

--help=false Print usage

-m, --memory= Memory limit

--memory-swap= Total memory (memory + swap), '-1' to disable swap

--no-cache=false Do not use cache when building the image构建过程中是否使用缓存

--pull=false Always attempt to pull a newer version of the image

-q, --quiet=false Suppress the verbose output generated by the containers

--rm=true Remove intermediate containers after a successful build 是否移除中间过程的容器

-t, --tag= Repository name (and optionally a tag) for the image 镜像命名

--ulimit=[] Ulimit options

提交到私有镜像源命令 docker push registy-server:5000/ubuntu

所有的 Dockerfile 命令格式都是: INSTRUCTION arguments

#作为注释

1. dcokerfile语法总结

**(a)FROM**

语法：FROM <image>或FROM <image>:<tag>

说明：这个设置基本的镜像，为后续的命令使用，所以应该作为Dockerfile的第一条指令。如果指定的镜像不存在默认会自动从Docker Hub上下载。

**(b)MAINTAINER**

语法：MAINTAINER <name>

说明：MAINTAINER命令用来指定维护者的姓名和联系方式，如：“xiaohei，wangp@dtdream.com”

**(c)RUN**

语法：RUN <command>或RUN ["executable", "param1", "param2" ... ]

说明：RUN命令会在上面FROM指定的镜像里执行任何命令，然后提交(commit)结果，提交的镜像会在后面继续用到。通常这里会执行一些安装软件的命令，如apt-get install –y python等

**(d)CMD**

语法：

CMD ["executable", "param1", "param2"] #将会调用exec执行，首选方式

CMD ["param1","param2"] #当使用ENTRYPOINT指令时，为该指令传递默认参数

CMD <command> [ <param1>|<param2> ] #将会调用/bin/sh -c执行

说明：CMD指令中指定的命令会在镜像运行时执行，在Dockerfile中只能存在一个，如果使用了多个CMD指令，则只有最后一个CMD指令有效。当出现ENTRYPOINT指令时，CMD中定义的内容会作为ENTRYPOINT指令的默认参数，也就是说可以使用CMD指令给ENTRYPOINT传递参数。

注意：RUN和CMD都是执行命令，他们的差异在于RUN中定义的命令会在执行[docker](http://www.server110.com/docker/) build命令创建镜像时执行，而CMD中定义的命令会在执行[docker](http://www.server110.com/docker/) run命令运行镜像时执行，另外使用第一种语法也就是调用exec执行时，命令必须为绝对路径。

**(e)ENTRYPOINT**

语法：

ENTRYPOINT cmd param1 param2 ... #将会调用/bin/sh -c执行

ENTRYPOINT ["cmd","param1","param2"...] #将会调用exec执行，首选方式

说明：ENTRYPOINT指令中指定的命令会在镜像运行时执行，在Dockerfile中只能存在一个，如果使用了多个ENTRYPOINT指令，则只有最后一个指令有效。ENTRYPOINT指令中指定的命令(exec执行的方式)可以通过docker run来传递参数，例如docker run <images> -l启动的容器将会把-l参数传递给ENTRYPOINT指令定义的命令并会覆盖CMD指令中定义的默认参数(如果有的话)，但不会覆盖该指令定义的参数，例如ENTRYPOINT ["ls","-a"]，CMD ["/etc"],当通过docker run <image>启动容器时该容器会运行ls -a /etc命令，当使用docker run <image> -l启动时该容器会运行ls -a -l命令，-l参数会覆盖CMD指令中定义的/etc参数。

注意：当使用ENTRYPOINT指令时生成的镜像运行时只会执行该指令指定的命令。当出现ENTRYPOINT指令时CMD指令只可能(当ENTRYPOINT指令使用exec方式执行时)被当做ENTRYPOINT指令的参数使用，其他情况则会被忽略。ENTRYPOINT 命令设置在容器启动时执行的这个命令需要挂住bash，否则就会立即退出。

**(f) USER**

语法：USER <username>

说明：USER指令用于设置用户或uid来运行生成的镜像和执行RUN指令，指定运行时的用户，即执行启动命令的用户，默认为root，有些软件可以通过-u参数来指定

**(g) EXPOSE**

语法：EXPOSE <port> [<port>...]

说明：设置一个端口在运行的镜像中暴露在外

**(h) ENV**

语法：ENV <key> <value>

说明：用于设置环境变量,后续的RUN命令都可以使用,某些软件需要一些环境变量才可以运行，这与通过docker run –e参数来配置的环境变量不同是在构建和运行时都可以使用的环境变量，而-e参数只能在运行时使用。最好不要定义那些可能和系统预定义的环境变量冲突的名字，否则可能会产生意想不到的结果。

**(i) ADD&COPY**

语法：ADD <src> <dest>& COPY <src> <dest>

说明: <src> 是相对被构建的源目录的相对路径，可以是文件或目录的路径，也可以是一个远程的文件url,<dest> 是container中的绝对路径

注意：

①如果执行docker build – < somefile即通过标准输入来创建时，ADD指令只支持url方式，另外如果url需要认证，则可以通过RUN wget …或RUN curl …来完成，ADD指令不支持认证。

②<src>路径必须与Dockerfile在同级目录或子目录中，例如不能使用ADD ../somepath，因为在执行docker build时首先做的就是把Dockerfile所在目录包含子目录发送给docker的守护进程。

③如果<src>是一个url且<dest>不是以”/“结尾，则会下载文件并重命名为<dest>。

④如果<src>是一个url且<dest>以“/”结尾，则会下载文件到<dest>/<filename>，url必须是一个正常的路径形式，“http://example.com”像这样的url是不能正常工作的。

⑤如果<src>是一个本地的压缩包且<dest>是以“/”结尾的目录，则会调用“tar -x”命令解压缩，如果<dest>有同名文件则覆盖，但<src>是一个url时不会执行解压缩。

⑤用法与ADD相同，不过<src>不支持使用url，所以在使用docker build – < somefile时该指令不能使用。

**(j) VOLUME**

语法：VOLUME ["samepath"]

说明：VOLUME指令用来设置一个挂载点，可以用来让其他容器挂载以实现数据共享或对容器数据的备份、恢复或迁移。

**(k) WORKDIR**

语法：WORKDIR /path/to/workdir

说明：WORKDIR指令用于设置Dockerfile中的RUN、CMD和ENTRYPOINT指令执行命令的工作目录(默认为/目录)，该指令在Dockerfile文件中可以出现多次，如果使用相对路径则为相对于WORKDIR上一次的值，例如WORKDIR /data，WORKDIR logs，RUN pwd最终输出的当前目录是/data/logs。

**(l) ONBUILD**

语法：ONBUILD [INSTRUCTION]

说明：ONBUILD指令用来设置一些触发的指令，用于在当该镜像被作为基础镜像来创建其他镜像时(也就是Dockerfile中的FROM为当前镜像时)执行一些操作，ONBUILD中定义的指令会在用于生成其他镜像的Dockerfile文件的FROM指令之后被执行，上述介绍的任何一个指令都可以用于ONBUILD指令，可以用来执行一些因为环境而变化的操作，使镜像更加通用。

注意：ONBUILD中定义的指令在当前镜像的build中不会被执行。可以通过查看docker inspeat <image>命令执行结果的OnBuild键来查看某个镜像ONBUILD指令定义的内容。ONBUILD中定义的指令会当做引用该镜像的Dockerfile文件的FROM指令的一

#### 示例

目录结构：

images

├── apache-tomcat-8.0.30.tar.gz

├── Dockerfile

├── jre-8u65-linux-x64.tar.gz

└── tomcat-users.xml

DockerFile:

**FROM ubuntu:14.04**

**MAINTAINER xiaohei "wangp@dtdream.com"**

**COPY . /tmp**

**ENV JAVA\_HOME /opt/jre1.8.0\_65**

**ENV CATALINA\_HOME /opt/apache-tomcat-8.0.30**

**ENV PATH $PATH:$JAVA\_HOME/bin:$CATALINA\_HOME/bin**

**ENV TOMCAT\_VERSION 8.0.28**

**ENV JAVA\_VERSION 8u65**

**RUN tar -xf /tmp/apache-tomcat-8.0.30.tar.gz -C /opt/ \**

**&& tar -xf /tmp/jre-8u65-linux-x64.tar.gz -C /opt/ \**

**&& rm -rf /tmp/\* $JAVA\_HOME/man**

**ADD tomcat-users.xml $CATALINA\_HOME/conf/**

**EXPOSE 8080**

**VOLUME $CATALINA\_HOME/webapps**

**CMD $CATALINA\_HOME/bin/startup.sh && tail -f /dev/null**

# 三、疑问和总结

1. Kubernetes如何指定私有镜像？

可以通过多种方式指定私有镜像

1. 使用谷歌仓库（略）
2. 配置节点向私有仓库鉴权

所有pods能够读取任何配置好的私有仓库

需要集群管理员配置节点

1. 缓存镜像

所有pods能够使用任何一个几点上缓存的镜像

需要管理员进入所有节点设置

1. 在pod上设置镜像拉取安全策略

只有提供自己的key的pod能够访问私有仓库

1. 如何制作自定义镜像并上传到私有镜像源？

制作镜像方法一：使用docker build创建镜像需要编写Dockerfile.执行命令必须能挂起控制台

制作镜像方法二：docker commit 是往版本控制系统里提交一次变更。使用这种方式制作镜像，本质上是运行一个基础镜像，然后在基础镜像上进行软件安装和修改。最后再将改动提交到版本系统中。

将本地镜像tag上私有仓库标签在push到私有仓库

docker tag tomcat8:jre8 10.158.240.100:5000/tomcat8:jre8

docker push 10.158.240.100:5000/tomcat8:jre8

1. 如何加速Docker Image pull速度？

1、学会使用并优化regisry

2、如果要提高拉取镜像的速度，建议减小镜像的大小，优化网络，完全使用局域网的镜像

1. 如何做高可靠性？

registy镜像可以配置https

<http://www.linuxidc.com/Linux/2015-04/116775p3.htm>

<http://segmentfault.com/a/1190000000801162>

5、镜像存储方案？（ceph、nfs或者oss）

根据需求配置registry，目前支持多种方式

Image pull push

consul ok ok

etcd ok ok

logstash ok ok

memcached ok ok

elasticsearch ok ok

rabbitmq ok ok

redis ok ok

nats ok ok

mongodb ok ok

mysql ok ok

neo4j ok ok

arangodb ok ok

couchdb ok ok

rethinkdb ok ok