

CentOS7.2 – Docker 容器虚拟化

Docker 认知了解:

Docker 理念: 一次封装,到处运行

解决了运行环境和配置问题软件容器,方便做持续集成并有助于整体发布的容器虚拟化技术

虚拟机的缺点: 1.资源占用多 2.冗余步骤多 3.启动慢

比较 Docker 和 传统虚拟化 方式的不同之处:

*传统虚拟机技术是虚拟出一套硬件后,在其上运行一个完整操作系统,在该系统上再运行所需应用进程

*而容器内的应用进程直接运行于宿主的内核,容器内没有自己的内核,而且也没有进行硬件虚拟,因此容器要比传统虚拟机更为轻便

*每个容器之间互相隔离,每个容器有自己的文件系统,容器之间进程不会相互影响,能区分计算资源

使用 Docker 容器虚拟化技术的好处:

1. 更快速的应用交付和部署
2. 更便捷的升级和扩缩容
3. 更简单的系统运维
4. 更高效的计算资源利用

Docker 网址:

docker 官方网站: http://www.docker.com docker 中文网站: https://www.docker-cn.com/ docker Hub 仓库: https://hub.docker.com/
--

Docker 三要素: 仓库(repository) 镜像(image) 容器(container)

仓库(repository)是集中存放镜像文件的场所

仓库(repository)和仓库注册服务器(registry)是有区别的.仓库注册服务器上往往存放着多个仓库,每个仓库中又包含了多个镜像,每个镜像有不同的标签(tag)

仓库分为公开仓库(public)和私有仓库(private)两种形式

最大的公开仓库是:Docker Hub(<https://hub.docker.com/>),存放了数量庞大的镜像供用户下载,国内的公开仓库包括阿里云、网易云等

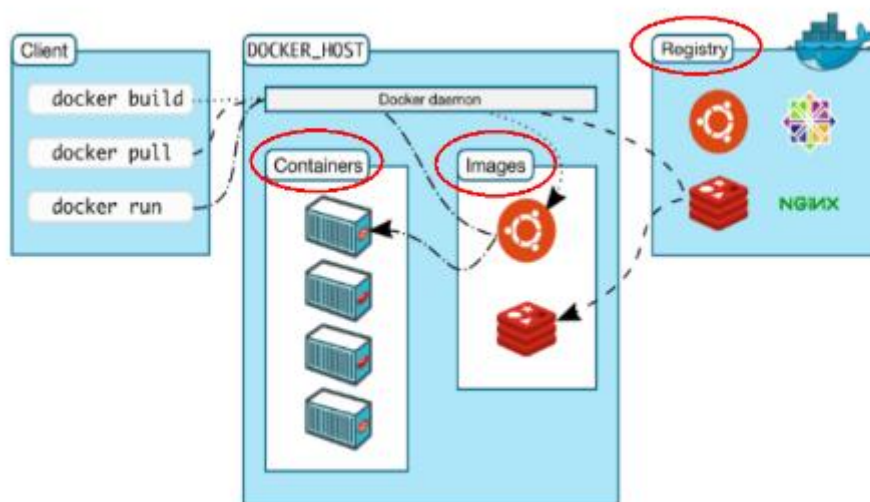
Docker 镜像(image)就是一个只读的模板,镜像可以用来创建 Docker 容器,一个镜像可以创建很多容器

Docker 容器(container)就是独立运行的一个或一组应用,容器是用镜像创建的运行实例

容器可以被启动、开始、停止、删除,每个容器都是相互隔离的,保证安全的平台

可以把容器看做是一个简易版的 Linux 环境(包括 root 用户权限、进程空间、用户空间和网络空间等)和运行在其中的应用程序

Docker 架构图:



容器与镜像的关系类似与面向对象编程中的对象与类:

Docker	面向对象
容器	对象
镜像	类

正确的理解仓储/镜像/容器这几个概念:

Docker 本身是一个容器运行载体或称之为管理引擎.我们把应用程序和配置依赖打包好形成一个可交付的运行环境,这个打包好的运行环境就是 **image** 镜像文件,只有通过这个镜像文件才能生成 Docker 容器,image 文件可以看作是容器的模板,Docker 根据 image 文件生成容器的实例,同一个 image 文件,可以生成多个同时运行的容器实例

- * 镜像文件生成的容器实例,本身也是一个文件,称为镜像文件
- * 一个容器运行一种服务,当我们需要的时候,就可以通过 **docker** 客户端创建一个对应的运行实例,也就是我们的容器
- * 至于仓储,就是放了一堆镜像的地方,我们可以把镜像发布到仓储中,需要的时候从仓储中拉下来就可以了

Docker 安装测试:

安装 docker:

参考手册: <https://docs.docker-cn.com/engine/installation/linux/docker-ce/centos/#prerequisites>

安装准备:

```
yum -y install gcc
```

```
yum -y install gcc-c++
```

```
yum install -y yum-utils device-mapper-persistent-data lvm2
```

```
# yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

yum-config-manager

--add-repo

http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo

yum makecache fast

安装命令: yum -y install docker-ce

启动 **docker**:

启动: systemctl start docker

测试是否安装成功:

测试: docker version

配置镜像加速器:

创建目录: mkdir -p /etc/docker

创建文件: vi /etc/docker/daemon.json 并添加如下内容

```
# 网易云
{
  "registry-mirrors": ["http://hub-mirror.c.163.com"]
}
# 阿里云
{
  "registry-mirrors": ["https://1qlfqh72.mirror.aliyuncs.com "]
}
```

重新加载: systemctl daemon-reload

重新启动: systemctl restart docker

检测镜像加速是否生效: ps -ef | grep docker

```
[root@CentosServer ~]# ps -ef | grep docker
root      2871      1   0 10:46 ?        00:00:01 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
root      3066    2486   0 10:49 pts/0    00:00:00 grep  --color=auto docker
```

重启 **docker**:

重启: systemctl restart docker 或 service docker restart

卸载 **docker**:

停止: systemctl stop docker

卸载: yum -y remove docker-ce

删除: rm -rf /var/lib/docker

运行状态 docker:

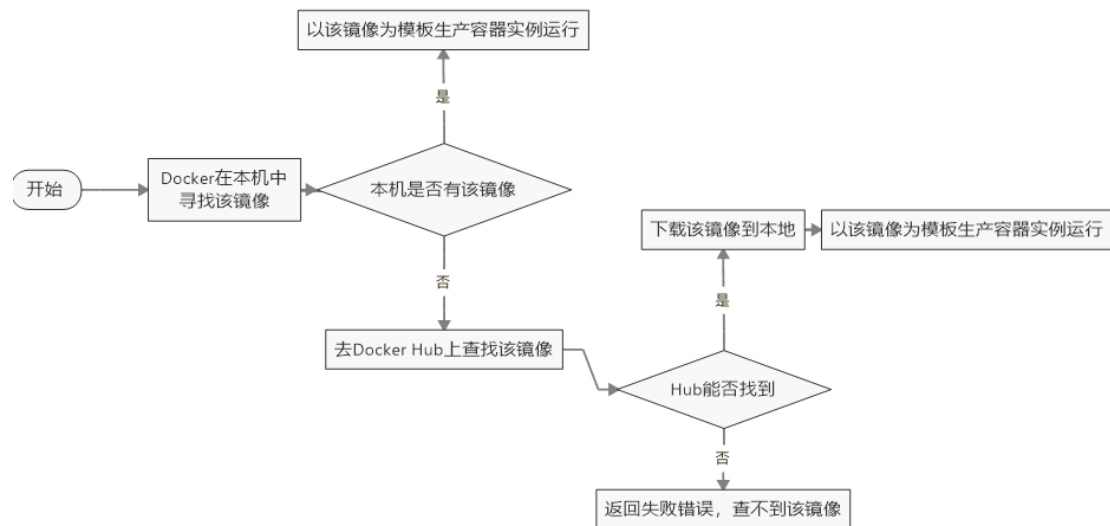
状态: `systemctl status docker`

测试运行 hello-world:

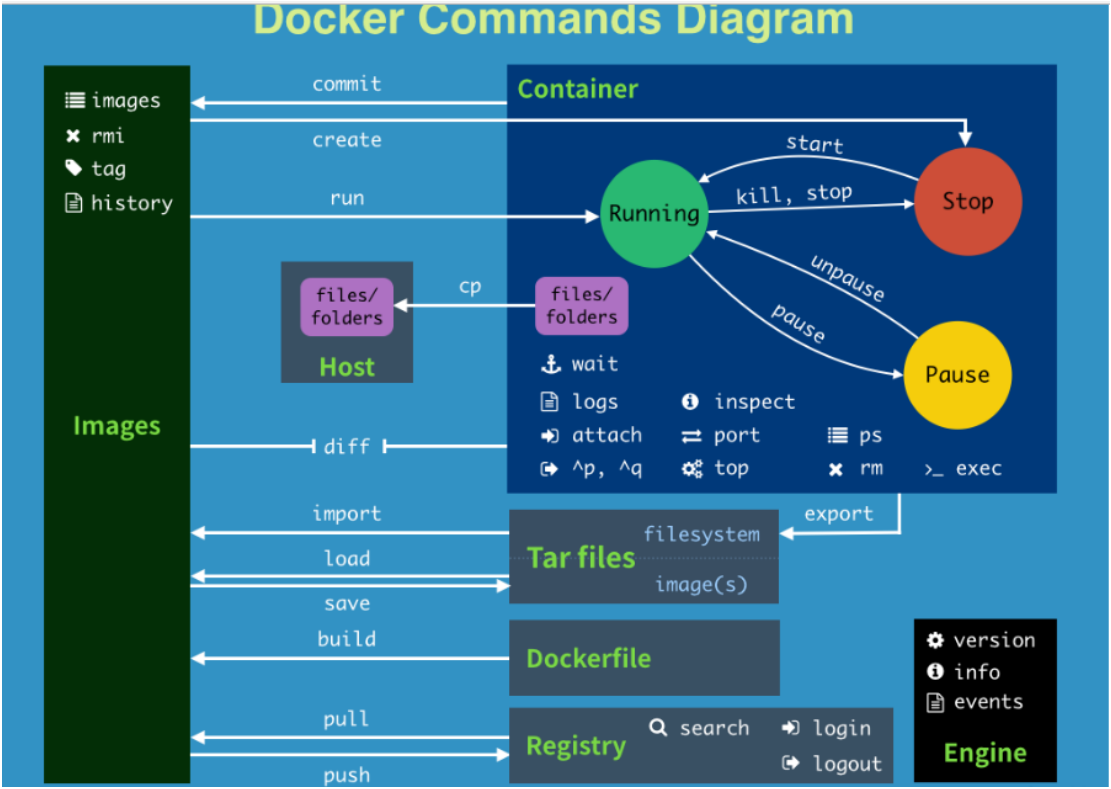
命令: `docker run hello-world`

```
[root@CentosServer ~]# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:c3b4ada4687bbaa170745b3e4dd8ac3f194ca95b2d0518b417fb47e5879d9b5f
Status: Downloaded newer image for hello-world:latest
```

run 都干了什么操作:



Docker 常用命令:



帮助命令:

Docker 验证: docker version
Docker 信息: docker info
Docker 辅助命令查询: docker -help

Docker 镜像命令:

列出本机镜像: docker images

命令: docker images

```
[root@CentosServer ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello-world          latest              fce289e99eb9       9 months ago       1.84kB
```

REPOSITORY	表示镜像的仓库源
TAG	镜像的标签
IMAGE ID	镜像 ID
CREATED	镜像创建时间

SIZE	镜像大小
------	------

同一仓库源可以有多个 TAG,代表这个仓库源的不同个版本,我们使用 REPOSITORY:TAG 来定义不同的镜像

如果不指定一个镜像的版本标签,Docker 将默认使用 Xxx:latest 镜像(最新版本)

参数信息说明:

docker images -a	-a: 列出本地所有的镜像(含中间映像层)
docker images -q	-q: 只显示镜像 ID
docker images --digests	--digests: 显示镜像的摘要信息
docker images --no-trunc	--no-trunc: 显示完整的镜像信息

```
[root@CentosServer ~]# docker images -a
REPOSITORY      TAG              IMAGE ID          CREATED           SIZE
hello-world      latest          fce289e99eb9     9 months ago     1.84kB
```

```
[root@CentosServer ~]# docker images -q
fce289e99eb9
```

```
[root@CentosServer ~]# docker images --digests
REPOSITORY      TAG              SIZE              DIGEST              IMAGE ID
hello-world      latest          1.84kB            sha256:c3b4ada4687bbaa170745b3e4dd8ac3f194ca95b2d0518b417fb47e5879d9b5f  fce289e99eb9
9 months ago
```

查找某个镜像: docker search xxx 镜像名称

命令: docker search tomcat

```
[root@CentosServer ~]# docker search tomcat
NAME                DESCRIPTION                STARS                OFFICIAL            AUTOMATED
tomcat              Apache Tomcat is an open source implementati... 2544                [OK]
tomcat              Apache Tomcat is an open source implementati... 69                  [OK]
dordoka/tomcat      Ubuntu 14.04, Oracle JDK 8 and Tomcat 8 base... 53                  [OK]
bitnami/tomcat      Bitnami Tomcat Docker Image 29                  [OK]
kubeguide/tomcat-app Tomcat image for Chapter 1 28
consol/tomcat-7.0    Tomcat 7.0.57, 8080, "admin/admin" 16                  [OK]
```

参数说明:

docker search --s 10 tomcat	-s: 列出收藏数不小于指定值的镜像
docker search tomcat --no-trunc	--no-trunc: 显示完整的镜像描述
docker search tomcat --automated	--automated: 只列出 automated build 类型的镜像

```
[root@CentosServer ~]# docker search -s 10 tomcat
Flag --stars has been deprecated, use --filter=stars=3 instead
NAME                DESCRIPTION                STARS                OFFICIAL            AUTOMATED
tomcat              Apache Tomcat is an open source implementati... 2544                [OK]
tomcat              Apache Tomcat is an open source implementati... 69                  [OK]
dordoka/tomcat      Ubuntu 14.04, Oracle JDK 8 and Tomcat 8 base... 53                  [OK]
bitnami/tomcat      Bitnami Tomcat Docker Image 29                  [OK]
```

```
[root@CentosServer ~]# docker search tomcat --no-trunc
NAME                STARS                OFFICIAL            AUTOMATED
tomcat              2544                [OK]
tomcat              69                  [OK]
dordoka/tomcat      53                  [OK]
bitnami/tomcat      29                  [OK]
DESCRIPTION
Apache Tomcat is an open source implementation of the Java Servlet and JavaServer Pages technologies
Apache TomEE is an all-Apache Java EE certified stack where Apache Tomcat is top dog.
Ubuntu 14.04, Oracle JDK 8 and Tomcat 8 based docker container.
Bitnami Tomcat Docker Image
```

```
[root@CentosServer ~]# docker search tomcat --automated
Flag --automated has been deprecated, use --filter=is-automated=true instead
NAME                DESCRIPTION                STARS                OFFICIAL            AUTOMATED
dordoka/tomcat      Ubuntu 14.04, Oracle JDK 8 and Tomcat 8 base... 53                  [OK]
bitnami/tomcat      Bitnami Tomcat Docker Image 29                  [OK]
consol/tomcat-7.0    Tomcat 7.0.57, 8080, "admin/admin" 16                  [OK]
```

下载某个镜像: docker pull xxx 镜像名称[:TAG]

命令: docker pull tomcat 等价于: docker pull tomcat:latest(最新版)

```
[root@CentosServer ~]# docker pull tomcat
Using default tag: latest
latest: Pulling from library/tomcat
9a0b0ce99936: Downloading [=====>] 27.3MB/45.38MB
db3b6004c61a: Download complete
f8f075920295: Download complete
6ef14aff1139: Downloading [=====>] 18.04MB/50.07MB
962785d3b7f9: Download complete
631589572f9b: Download complete
c55a0c6f4c7b: Downloading [==>] 4.2MB/104.2MB
379605d88e88: Waiting
e056aa10ded8: Waiting
6349a1c98d85: Waiting
```

```
[root@CentosServer ~]# docker pull tomcat
Using default tag: latest
latest: Pulling from library/tomcat
9a0b0ce99936: Pull complete
db3b6004c61a: Pull complete
f8f075920295: Pull complete
6ef14aff1139: Pull complete
962785d3b7f9: Pull complete
631589572f9b: Pull complete
c55a0c6f4c7b: Pull complete
379605d88e88: Pull complete
e056aa10ded8: Pull complete
6349a1c98d85: Pull complete
Digest: sha256:77e41dbdf7854f03b9a933510e8852c99d836d42ae85cba4b3bc04e8710dc0f7
Status: Downloaded newer image for tomcat:latest
docker.io/library/tomcat:latest
```

删除镜像: docker rmi -f xxx 镜像名称[:TAG]/镜像 ID

命令: docker rmi -f xxx 镜像名称[:TAG] /镜像 ID

注: docker rmi -f tomcat 默认是删除 latest 标签的 tomcat: 即 docker rmi -f tomcat:latest

删除单个: docker rmi -f 镜像名称[:TAG]/镜像 ID

```
[root@CentosServer ~]# docker rmi -f hello-world
Untagged: hello-world:latest
Untagged: hello-world@sha256:c3b4ada4687bbaa170745b3e4dd8ac3f194ca95b2d0518b417fb47e5879d9b5f
Deleted: sha256:fce289e99eb9bca977dae136f8e2a82b6b7d4c372474c9235adc1741675f587e
```

```
[root@CentosServer ~]# docker rmi -f fce289e99eb9
Untagged: hello-world:latest
Untagged: hello-world@sha256:c3b4ada4687bbaa170745b3e4dd8ac3f194ca95b2d0518b417fb47e5879d9b5f
Deleted: sha256:fce289e99eb9bca977dae136f8e2a82b6b7d4c372474c9235adc1741675f587e
```

删除多个: docker rmi -f 镜像名称 1[:TAG]/镜像 ID 镜像名称 2[:TAG]/镜像 ID

```
[root@CentosServer ~]# docker rmi -f hello-world redis
Untagged: hello-world:latest
Untagged: hello-world@sha256:c3b4ada4687bbaa170745b3e4dd8ac3f194ca95b2d0518b417fb47e5879d9b5f
Deleted: sha256:fce289e99eb9bca977dae136f8e2a82b6b7d4c372474c9235adc1741675f587e
Untagged: redis:latest
Untagged: redis@sha256:fe80393a67c7058590ca6b6903f64e35b50fa411b0496f604a85c526fb5bd2d2
Deleted: sha256:de25a81a5a0b6ff26c82bab404fff5de5bf4bbbc48c833412fb3706077d31134
Deleted: sha256:b39d98a508cb9ec9e080e09606005660caa3d3d8dd00083e333e6536114d531b
Deleted: sha256:8232e1dcaa8e9392b2532e88f908e2d17a59203d1bc873d0f8cdbe68dfaeb4f
Deleted: sha256:9064b95e7336eae34e78f6144214770b77319ddd2a185e8259becbc48a124a1
Deleted: sha256:9be905466faa79aaf23c765addb691915a3a6f44e214cfde3707d24f99d304d0
Deleted: sha256:c2aceb594f2f81781fb2afc198bde3f54de46a05666e33eab254e56b3396770f
```

删除全部: docker rmi -f \$(docker images -qa)

```
[root@CentosServer MyDocker]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
3fb3e0997a00       centos             "/bin/bash"        13 seconds ago     Up 10 seconds      80/tcp             naughty_
99f53c3c55e9       test/centos:1.1    "/bin/sh -c /bin/bash"  19 minutes ago     Up 19 minutes      80/tcp             dazzling
[root@CentosServer MyDocker]# docker rm -f $(docker ps -qa)
3fb3e0997a00
99f53c3c55e9
[root@CentosServer MyDocker]#
```

镜像变更历史:

自定义镜像: docker history 自定义镜像名称/自定义镜像 ID

如: docker history test/centos:1.1

```
[root@CentosServer MyDocker]# docker history test/centos:1.1
IMAGE               CREATED             CREATED BY          SIZE                COMMENT
c2174ea6504d        3 hours ago        /bin/sh -c #(nop)  CMD ["/bin/sh" "-c" "/bin...  0B
5f3c17d9c1fa        3 hours ago        /bin/sh -c #(nop)  EXPOSE 80             0B
e7581062dd78        3 hours ago        /bin/sh -c yum -y install net-tools  14.3MB
a7669129f258        3 hours ago        /bin/sh -c yum -y install vim        55.3MB
4b8aeab8ac6d        3 hours ago        /bin/sh -c #(nop)  WORKDIR /tmp          0B
32a19bd69e1b        3 hours ago        /bin/sh -c #(nop)  ENV mypath=/tmp        0B
0f3e07c0138f        4 weeks ago        /bin/sh -c #(nop)  CMD ["/bin/bash"]     0B
<missing>            4 weeks ago        /bin/sh -c #(nop)  LABEL org.label-schema.sc...  0B
<missing>            4 weeks ago        /bin/sh -c #(nop)  ADD file:d6fdacc1972df524a...  220MB
[root@CentosServer MyDocker]#
```

Docker 容器命令: ★

命令: docker run [options] 镜像名称/镜像 ID

参数说明: options

-i(交互)	以交互模式运行容器,通常与 -t 同时使用	
-t(终端)	为容器重新分配一个伪输入终端,通常与 -i 同时使用	
--name="容器新名字"	为容器指定一个名称	
-d	后台运行容器,并返回容器 ID,也即启动守护式容器	
-P(大写)	随机端口映射(随机分配端口)	如: docker run -it -P tomcat
-p(小写)	指定主机端口:镜像的容器端口	如: docker run -it -p 8888:8080 tomcat

创建容器并启动容器:

创建并启动容器: 以终端交互式方式创建容器并启动容器

命令: docker run -it centos 或 docker run -it 0f3e07c0138f

```
[root@CentosServer ~]# docker run -it centos
[root@66a3a8b38d04 /]#
```

创建并启动容器: 以终端交互式方式创建容器并启动容器,且给容器取个别名

命令: docker run -it --name="MyCentOS" centos

```
[root@CentosServer ~]# docker run -it --name="MyCentOS" centos
[root@1c1540151c37 /]#
```


创建并启动容器: 守护进程方式创建容器并启动容器

命令: docker run -d centos

命令: docker run -d -p 8888:8080 tomcat

```
[root@CentosServer ~]# docker run -d centos
8d59bbfec7475dd1cb2c63cb08897f6d1a2026e4843bf35bbda1fb9934bfc123
[root@CentosServer ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
[root@CentosServer ~]#
```

创建并启动容器:(大 p) 随机主机端口(镜像的容器使用默认端口)

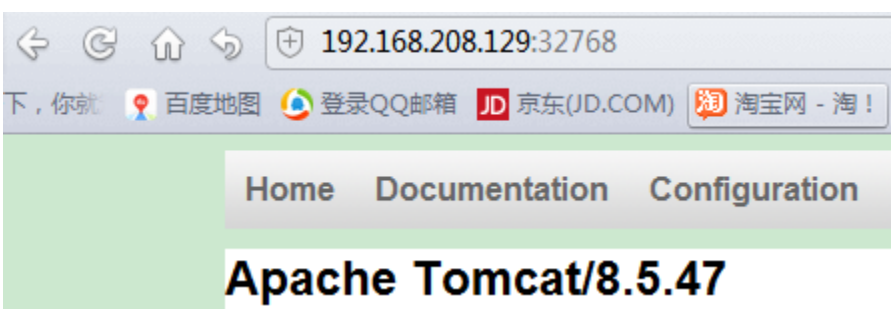
命令: docker run -it -P tomcat

```
[root@CentosServer ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
[root@CentosServer ~]# docker run -it -P tomcat
Using CATALINA_BASE: /usr/local/tomcat
Using CATALINA_HOME: /usr/local/tomcat
Using CATALINA_TMPDIR: /usr/local/tomcat/temp
Using JRE_HOME: /usr/local/openjdk-8
```

重新再次打开一个终端查询: docker ps

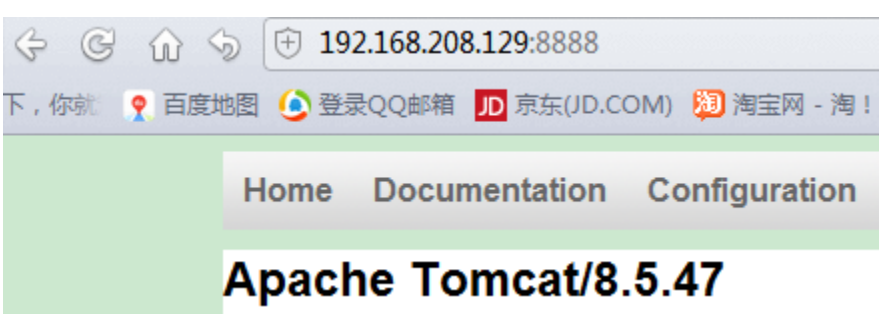
```
[root@CentosServer ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
6671576a09f7       tomcat             "catalina.sh run"   26 seconds ago     Up 23 seconds      0.0.0.0:32768->8080/tcp   naughty_feynman
```

根据查询到端口 32768 访问 tomcat: [http://IP 地址:32768](http://IP地址:32768)



创建并启动容器:(小 p) 指定主机端口及镜像的容器端口,以便外部访问

命令: docker run -it -p 8888:8080 tomcat



列出当前运行的容器:

命令: docker ps [options]

参数说明: options

docker ps -a	-a: 列出当前所有正在运行的容器 + 历史上运行过的
docker ps -l	-l: 显示最近创建的容器(即上一个容器)

docker ps -n 数字	-n: 显示最近 n 个创建的容器
docker ps -q	-q: 静默模式,只显示容器 ID 编号
docker ps --no-trunc	--no-trunc: 不截断输出

```

[root@CentosServer ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
3c2877b3fa59       0f3e07c0138f      "/bin/bash"        8 minutes ago       Up 8 minutes                sad_ishizaka
66a3a8b38d04       centos             "/bin/bash"        12 minutes ago      Up 11 minutes             condensing
44ca44157470       0f3e07c0138f      "/bin/bash"        15 minutes ago      Up 15 minutes             gifted_meitne

```

退出容器:

容器停止退出: exit

容器不停止退出: ctrl + p + q 可以重新再进入容器

启动容器:

命令: docker start 容器 ID/容器名称

如: docker start 66a3a8b38d04

```

[root@CentosServer ~]# docker start 66a3a8b38d04
66a3a8b38d04

```

重启容器:

命令: docker restart 容器 ID/容器名称

如: docker restart 66a3a8b38d04

```

[root@CentosServer ~]# docker restart 66a3a8b38d04
66a3a8b38d04

```

停止容器:

命令: docker stop 容器 ID/容器名称

如: docker stop 66a3a8b38d04

```

[root@CentosServer ~]# docker stop 66a3a8b38d04
66a3a8b38d04

```

强制停止运行容器:

命令: docker kill 容器 ID/容器名称

```
[root@CentosServer ~]# docker kill 1c1540151c37 1c1540151c37
```

删除已停止的容器:

删除一个已停止的容器: `docker rm 7371c90b7c8d`

删除多个已停止的容器: `docker rm 7371c90b7c8d 6571c90b7c8d`

删除所有已停止的容器: `docker rm -f $(docker ps -aq)` 或 `docker ps -aq | xargs docker rm`

★容器重要★:

启动守护式容器:

命令: `docker run -d 镜像名称/镜像 ID`

命令: `docker run -d -p 8888:8080 tomcat`

```
[root@CentosServer ~]# docker run -d -p 7777:8080 tomcat
39b93243e02c491327906dd7eda3af60aa1f0e854ccc8fd6c5a81451dfdac41
[root@CentosServer ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
39b93243e02c        tomcat             "catalina.sh run"   8 seconds ago       Up 4 seconds        0.0.0.0:7777->8080/tcp   optim
a4c920cae8f6        tomcat             "catalina.sh run"   2 minutes ago       Up 2 minutes        8080/tcp              stoic
3b2458f8fda5        chendong/tomcat8:1.2 "catalina.sh run"   19 minutes ago      Up 19 minutes       0.0.0.0:8888->8080/tcp   funny
[root@CentosServer ~]#
```

查看容器的日志:

命令: `docker logs -t -f -till num 容器 ID/容器名称`

参数说明:

-t	是加入时间戳
-f	跟随最新的日志打印
--tail num	数字 显示最后多少条

查看容器内运行的进程:

命令: `docker top 容器 ID/容器名称`

```
[root@CentosServer ~]# docker top 4401623ac54b
UID                PID                PPID              C                STIME            TTY              TIME
root               13455             13438             0                09:34            pts/0            00:00:00
/bin/bash
[root@CentosServer ~]# docker top strange_matsumoto
UID                PID                PPID              C                STIME            TTY              TIME
root               13455             13438             0                09:34            pts/0            00:00:00
/bin/bash
```

查看容器内的内部细节:

命令: docker inspect 容器 ID/容器名称

进入正在运行的容器并以命令行交互:

命令:

docker exec -it 容器 ID /bin/bash	进入容器中打开新的终端,并且可以启动新的进程
docker exec -it 容器 ID ls -l /tmp	进入容器并执行命令,显示结果(不进入容器的终端)
docker attach 容器 ID	attach: 重新进入已启动容器的命令行终端,不会启动新的进程

```
[root@CentosServer ~]# docker ps
CONTAINER ID   IMAGE          COMMAND        CREATED        STATUS        PORTS        NAMES
4401623ac54b   0f3e07c0138f  "/bin/bash"    40 minutes ago Up 40 minutes        strange_matsu
moto
[root@CentosServer ~]# docker exec -it 4401623ac54b /bin/bash
[root@4401623ac54b /]# ls -l /tmp
total 8
-rwx----- 1 root root 1379 Sep 27 17:13 ks-script-0n44nrd1
-rwx----- 1 root root 671 Sep 27 17:13 ks-script-w6m6m_20
[root@4401623ac54b /]#

[root@CentosServer ~]# docker ps
CONTAINER ID   IMAGE          COMMAND        CREATED        STATUS        PORTS        NAMES
4401623ac54b   0f3e07c0138f  "/bin/bash"    42 minutes ago Up 42 minutes        strange_matsu
moto
[root@CentosServer ~]# docker exec -it 4401623ac54b ls -l /tmp
total 8
-rwx----- 1 root root 1379 Sep 27 17:13 ks-script-0n44nrd1
-rwx----- 1 root root 671 Sep 27 17:13 ks-script-w6m6m_20
[root@CentosServer ~]#

[root@CentosServer ~]# docker ps
CONTAINER ID   IMAGE          COMMAND        CREATED        STATUS        PORTS        NAMES
4401623ac54b   0f3e07c0138f  "/bin/bash"    22 minutes ago Up 22 minutes        strange_matsu
moto
[root@CentosServer ~]# docker attach 4401623ac54b
[root@4401623ac54b /]#
```

从容器内拷贝文件到主机上:

命令: docker cp 容器 ID/容器名称:容器内路径 目的主机路径

```
[root@CentosServer ~]# ll
总用量 4
-rw----- 1 root root 1172 9月 24 15:40 anaconda-ks.cfg
[root@CentosServer ~]# docker ps
CONTAINER ID   IMAGE          COMMAND        CREATED        STATUS        PORTS        NAMES
4401623ac54b   0f3e07c0138f  "/bin/bash"    49 minutes ago Up 49 minutes        strange_matsu
moto
[root@CentosServer ~]# docker cp 4401623ac54b:/tmp/ks-script-0n44nrd1 /root/
[root@CentosServer ~]# ll
总用量 8
-rw----- 1 root root 1172 9月 24 15:40 anaconda-ks.cfg
-rwx----- 1 root root 1379 9月 28 01:13 ks-script-0n44nrd1
[root@CentosServer ~]#
```

Docker 镜像操作: commit

镜像是一种轻量级,可执行的独立软件包,用来打包软件运行环境和基于运行环境开发的软件,它包含运行某个软件所需的所有内容,包括代码,运行时,库,环境变量和配置文件

UnionFS(联合文件系统):Unio 文件系统支持对文件系统的修改作为一次提交来一层层的叠加

Union 文件系统是 Docker 镜像的基础;镜像可以通过分层来进行继承,基于基础镜像(没有父镜像),可以制作各种具体的应用镜像

docker 镜像采用分层结构的好处:

最大的一个好处就是 - 共享资源

比如:有多镜像都从相同的 base 镜像构建而来,那么宿主机只需在磁盘上保存一份 base 镜像,同时内存中也只需加载一份 base 镜像,就可以为所有容器服务了;而且镜像的每一层都可以被共享

示例如下: 提交容器副本使之成为一个新的镜像:

命令: **docker commit -m="提交的描述信息" -a="作者" 容器 ID/容器名字 新镜像名称:[标签 TAG]**

1. 创建一个容器并启动容器,可以对容器的内容做修改:

使用大 P 参数随机主机的端口(镜像的容器使用默认端口),如: `docker run -it -P tomcat`

使用小 p 参数指定主机的端口以及镜像的容器端口,如: `docker run -it -p 8888:8080 tomcat`

...修改(略)...

2. 把修改过后的容器副本作为一个新的镜像文件:

命令: `docker commit -a="chen" -m="delete tomcat docs" 6671576a09f7 chendong/tomcat8:1.2`

```
[root@CentosServer ~]# docker commit -a="chen" -m="delete tomcat docs" 6671576a09f7 chendong/tomcat8:1.2
sha256:17638b4766398b5dc2f88f24c9ba2665fafa7ca8c526743fa8e95ecc7267ac6c
[root@CentosServer ~]# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
chendong/tomcat8	1.2	17638b476639	12 seconds ago	507MB
mongo	latest	191b28dbfefe	32 hours ago	363MB
nginx	latest	540a289bab6c	2 days ago	126MB
tomcat	latest	882487b8be1d	6 days ago	507MB
redis	latest	de25a81a5a0b	8 days ago	98.2MB
centos	latest	0f3e07c0138f	3 weeks ago	220MB
hello-world	latest	fce289e99eb9	9 months ago	1.84kB

```
[root@CentosServer ~]#
```

3.使用自定义镜像创建容器并启动:

注: TAG 标签一定要正确

以交互式(前台)的方式启动容器

命令: `docker run -it -p 8888:8080 chendong/tomcat8:1.2`

```
[root@CentosServer ~]# docker run -it -p 8888:8080 chendong/tomcat8:1.2
Using CATALINA_BASE:   /usr/local/tomcat
Using CATALINA_HOME:   /usr/local/tomcat
Using CATALINA_TMPDIR: /usr/local/tomcat/temp
Using JRE_HOME:        /usr/local/openjdk-8
```

以守护式(后台)的方式启动容器

命令: `docker run -d -p 6666:8080 chendong/tomcat8:1.2`

```
[root@CentosServer ~]# docker run -d -p 6666:8080 chendong/tomcat8:1.2
437699c96792857d58f1c4ea0b1cbec05aa6638e5df65c38e94eccfcbcc004de
[root@CentosServer ~]# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
437699c96792	chendong/tomcat8:1.2	"catalina.sh run"	13 seconds ago	Up 9 seconds	0.0.0.0:6666->8080/tcp	nifty
39b93243e02c	tomcat	"catalina.sh run"	3 minutes ago	Up 3 minutes	0.0.0.0:7777->8080/tcp	optim
istic_austin	tomcat	"catalina.sh run"	6 minutes ago	Up 6 minutes	8080/tcp	stoic
allen	chendong/tomcat8:1.2	"catalina.sh run"	23 minutes ago	Up 23 minutes	0.0.0.0:8888->8080/tcp	funny
3b2458f8fda5						
wiles						

```
[root@CentosServer ~]#
```

Docker 容器数据卷: ★

容器数据卷的作用:

- 1.容器的持久化
2. 容器间继承 + 共享数据

容器数据卷的特点:

- 1:数据卷可在容器之间共享或重用数据
- 2:数据卷中的更改可以直接生效
- 3:数据卷中的更改不会包含在镜像的更新中
- 4:数据卷的生命周期一直持续到没有容器使用它为止

容器数据卷的添加方式: (两种方式)

使用命令添加: -v

不带权限: 可读可写

添加命令: `docker run -it -v /宿主机绝对路径目录:/容器内的绝对路径目录 镜像名称/镜像 ID`
如: `docker run -it -v /HostData:/ContainerData centos`

容器 Container:

```
[root@CentosServer ~]# docker run -it -v /HostData:/ContainerData centos
[root@9adb2e066ad /]# ls
ContainerData bin dev etc home lib lib64 lost+found media mnt opt proc root run sbin srv sys tmp usr var
[root@9adb2e066ad /]#
```

宿主机 Host:

```
[root@CentosServer /]# ls
bin boot dev etc home HostData lib lib64 media mnt opt proc root run sbin srv sys tmp usr var
[root@CentosServer /]#
```

是否挂载成功:

命令: `docker inspect 容器 ID`

```
[root@CentosServer /]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS        NAMES
9adb2e066ad   centos    "/bin/bash"             3 minutes ago Up 3 minutes                agitated_mats
umoto

[root@CentosServer /]# docker inspect 9adb2e066ad
```

带权限: 只读不可写(针对容器)

添加命令: `docker run -it -v /宿主机绝对路径目录:/容器内的绝对路径目录:ro 镜像名称/镜像 ID`

如: `docker run -it -v /HostData:/ContainerData:ro centos`

注: Docker 挂载主机目录 Docker 访问出现 `cannot open directory .: Permission denied???`

解决办法:在挂载目录后多加一个 `--privileged=true` 参数即可

如: `docker run -it -v /HostData:/ContainerData --privileged=true centos`

使用文件添加: Dockerfile

1. 在宿主机的根目录下创建 MyDocker 目录: `mkdir MyDocker`

```
[root@CentosServer ~]# mkdir MyDocker
[root@CentosServer ~]# ls
bin boot dev etc home HostData lib lib64 media mnt MyDocker opt proc root run sbin srv sys tmp usr var
```

2. 在 Dockerfile 中使用 volume 指令给镜像添加一个或多个数据卷(粗略使用)

```
[root@CentosServer ~]# cd MyDocker/
[root@CentosServer MyDocker]# pwd
/MyDocker
[root@CentosServer MyDocker]# vi Dockerfile
[root@CentosServer MyDocker]# ll
总用量 4
-rw-r--r--. 1 root root 134 10月 25 16:47 Dockerfile
```

```
# volume test
FROM centos
VOLUME ["/dataVolumeContainer1", "/dataVolumeContainer2"]
CMD echo "finished,-----success1"
CMD /bin/bash
```

3. 根据 Dockerfile 文件使用 build 构建一个新的镜像

命令: `docker build -f /MyDocker/Dockerfile -t chen/centos .`

```
[root@CentosServer MyDocker]# docker build -f /MyDocker/Dockerfile -t chen/centos .
Sending build context to Docker daemon 2.048kB
Step 1/4 : FROM centos
--> 0f3e07c0138f
Step 2/4 : VOLUME ["/dataVolumeContainer1", "/dataVolumeContainer2"]
--> Running in 3673c379f1a8
Removing intermediate container 3673c379f1a8
--> 3d3be7a5051d
Step 3/4 : CMD echo "finished,-----success1"
--> Running in 26000562ffac
Removing intermediate container 26000562ffac
--> 83596c3fb76f
Step 4/4 : CMD /bin/bash
--> Running in ed4e4ce7dd56
Removing intermediate container ed4e4ce7dd56
--> 9c8b13743cd3
Successfully built 9c8b13743cd3
Successfully tagged chen/centos:latest
[root@CentosServer MyDocker]#
```

4. 以构建的新镜像创建容器并启动容器,查看容器卷是否添加成功

```
[root@CentosServer MyDocker]# docker run -it chen/centos
[root@934bde0f417 /]# ls
bin dataVolumeContainer2 etc lib lost+found mnt proc run srv tmp
dataVolumeContainer1 dev home lib64 media opt root sbin sys usr
[root@934bde0f417 /]#
```

5. 在容器数据卷中创建一个文件,查看是否与宿主机数据共享

```
[root@934bde0f417 /]# cd dataVolumeContainer1/
[root@934bde0f417 dataVolumeContainer1]# vi container.py
```

6. Dockerfile 文件中虽然未指定宿主机与容器卷的关联关系(但默认已经关联)

查看命令: docker inspect 934bde0f417

```
"Mounts": [
  {
    "Type": "volume",
    "Name": "9037661ce2ede7b0c1674ad26c5e221c75fa5b39287e347f1a29bf3d422196b9",
    "Source": "/var/lib/docker/volumes/9037661ce2ede7b0c1674ad26c5e221c75fa5b39287e347f1a29bf3d422196b9/_data",
    "Destination": "/dataVolumeContainer2",
    "Driver": "local",
    "Mode": "",
    "RW": true,
    "Propagation": ""
  },
  {
    "Type": "volume",
    "Name": "a5fa8874abbd960ab7e0bd87a30a136f6fee3c9730a23c1adf66f9c95c93761f",
    "Source": "/var/lib/docker/volumes/a5fa8874abbd960ab7e0bd87a30a136f6fee3c9730a23c1adf66f9c95c93761f/_data",
    "Destination": "/dataVolumeContainer1",
    "Driver": "local",
    "Mode": "",
    "RW": true,
    "Propagation": ""
  }
]
```

在进入宿主机中对默认的路径位置进行查看

```
[root@CentosServer ~]# cd /var/lib/docker/volumes/a5fa8874abbd960ab7e0bd87a30a136f6fee3c9730a23c1adf66f9c95c93761f/_data
[root@CentosServer _data]# ll
总用量 4
-rw-r--r--. 1 root root 24 10月 25 16:59 container.py
[root@CentosServer _data]#
```

容器与宿主机之间数据共享:

容器 Container:

```
[root@9addb2e066ad /]# cd ContainerData/
[root@9addb2e066ad ContainerData]# vi aaa.py
[root@9addb2e066ad ContainerData]#
```

宿主机 Host:

```
[root@CentosServer /]# cd HostData/
[root@CentosServer HostData]# ll
总用量 4
-rw-r--r--. 1 root root 21 10月 25 15:56 aaa.py
[root@CentosServer HostData]#
```

注:容器停止并退出,宿主机修改数据后是否共享??? **共享 示例如下:**

容器 Container:

1.容器停止并退出: exit

```
[root@9addb2e066ad ContainerData]# exit
exit
[root@CentosServer ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
[root@CentosServer ~]#
```

3.重新启动容器,并进入启动的容器:

docker start 9addb2e066ad

docker attach 9addb2e066ad


```
[root@CentosServer ~]# docker start 9addb2e066ad
9addb2e066ad
[root@CentosServer ~]# docker attach 9addb2e066ad
[root@9addb2e066ad /]# cd ContainerData/
[root@9addb2e066ad ContainerData]# ls -l
total 4
-rw-r--r--. 1 root root 37 Oct 25 07:59 aaa.py
-rw-r--r--. 1 root root 0 Oct 25 08:04 bbb.java
[root@9addb2e066ad ContainerData]#
```

宿主机 Host:

2.宿主机添加 bbb.java 文件

```
[root@CentosServer HostData]# ll
总用量 4
-rw-r--r--. 1 root root 37 10月 25 15:59 aaa.py
[root@CentosServer HostData]# touch bbb.java
[root@CentosServer HostData]# ll
总用量 4
-rw-r--r--. 1 root root 37 10月 25 15:59 aaa.py
-rw-r--r--. 1 root root 0 10月 25 16:04 bbb.java
[root@CentosServer HostData]#
```

容器与容器间数据传递共享:

结论: 容器之间配置信息的传递,数据卷的生命周期一直持续到没有容器使用它为止

示例如下:

1. 启动一个父容器(别名为 **chen01**):

`docker run -it --name chen01 chen/centos`

```
[root@CentosServer ~]# docker run -it --name chen01 chen/centos
[root@a105d2c5959e /]# ls
bin          dataVolumeContainer2  etc      lib      lost+found  mnt      proc      run      srv      tmp      var
dataVolumeContainer1  dev              home     lib64    media       opt      root     sbin     sys      usr
```

2. 在父容器数据卷中添加一个文件: `touch /dataVolumeContainer2/01.txt`

```
[root@a105d2c5959e dataVolumeContainer2]# touch 01.txt
[root@a105d2c5959e dataVolumeContainer2]# pwd
/dataVolumeContainer2
[root@a105d2c5959e dataVolumeContainer2]# ls
01.txt
```

3. 再次启动两个容器(别名分别为 **chen02,chen03**),继承父容器 chen01: `--volumes-from`

`docker run -it --name chen02 --volumes-from chen01 chen/centos`

`docker run -it --name chen03 --volumes-from chen01 chen/centos`

```
[root@CentosServer ~]# docker run -it --name chen02 --volumes-from chen01 chen/centos
```

```
[root@CentosServer ~]# docker run -it --name chen03 --volumes-from chen01 chen/centos
```

4. 分别在容器 chen02 和容器 chen03 中的添加文件:

如 chen02 `touch /dataVolumeContainer2/02.txt`

如 chen03 touch /dataVolumeContainer2/03.txt

5. 再次进入如容器(chen01),查看数据卷下的文件是否共享: true

```
[root@CentosServer ~]# docker attach a105d2c5959e
[root@a105d2c5959e dataVolumeContainer2]# ls
01.txt  02.txt  03.txt
```

Docker-Dockerfile 构建镜像详解:

Dockerfile 能做什么:

Dockerfile 是用来构建 **Docker** 镜像的构建文件,是由一系列命令和参数构成的脚本文件

Dockerfile 构建步骤:

1. 编写 Dockerfile 文件: `vi /xxx/Dockerfile`
2. 构建新的镜像文件: **docker build -f Dockerfile 文件路径 -t 新镜像名称:TAG 标签** .
如: `docker build - Dockerfile - chen/centos` .
3. 运行新镜像的容器: **docker run -it chen/centos**

Dockerfile 构建解析:

Dockerfile 基础知识:

1. 每条保留字指令都必须为大写字母且后面要跟随至少一个参数
2. 每条保留字指令按照从上到下,顺序执行
3. #表示 Dockerfile 文件注释内容
4. 每条保留字指令都会创建一个新的镜像层,并对镜像进行提交

Dockerfile 执行流程:

1. Docker 基于基础镜像运行一个容器
2. 执行一条保留字指令并对容器作出修改
3. 执行类似 `docker commit` 的操作提交一个新的镜像层
4. Docker 再基于刚提交的镜像运行一个新容器
5. 执行 Dockerfile 中的下一条保留字指令直到所有指令都执行完成

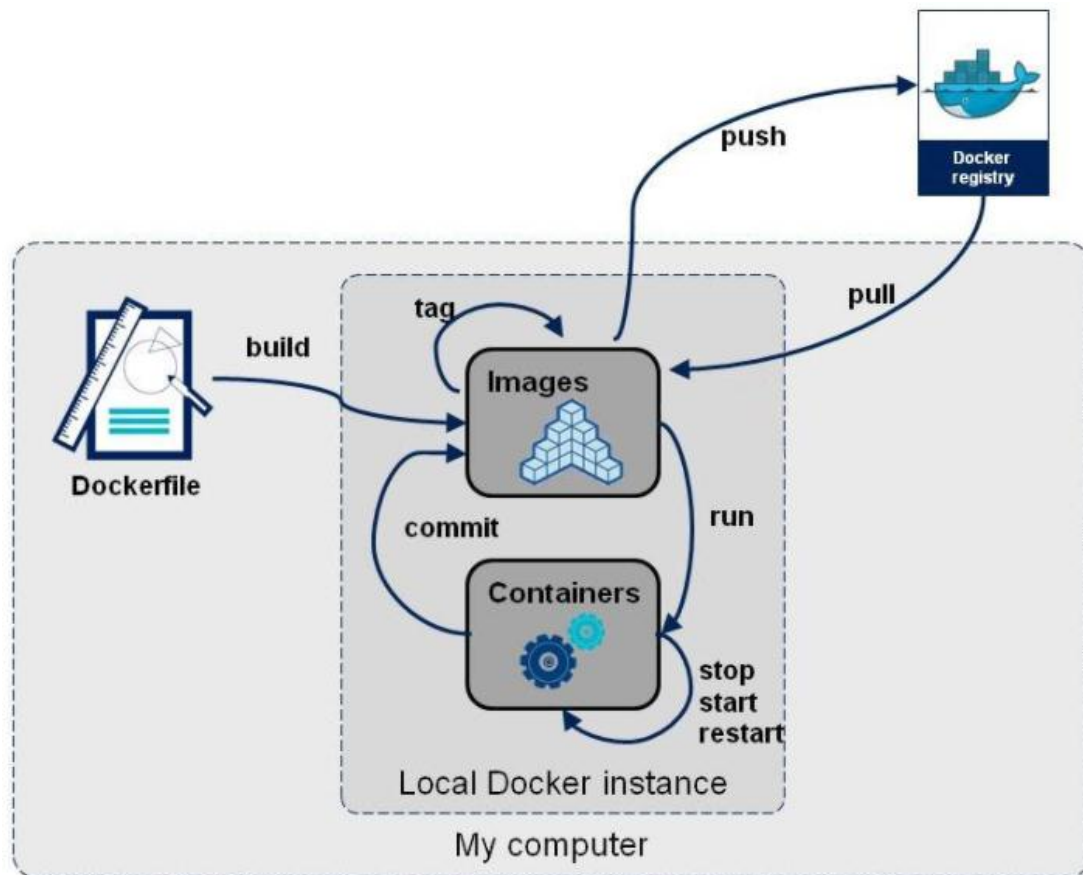
Dockerfile 体系结构:

Dockerfile 保留字指令:

Docker 的镜像基本上都是通过 Base(scratch)镜像安装和配置需要的软件构建出来的

FROM	(from): 基础镜像,当前构建的新镜像是基于哪个镜像的
MAINTAINER	(maintainer): 新镜像维护者的姓名和邮箱地址
RUN	(run): 容器构建时需要运行的命令
EXPOSE	(expose): 容器运行时对外暴露的端口
WORKDIR	(workdir): 容器创建后,终端默认登陆的进来工作目录路径,一个落脚点
ENV	(env): 构建镜像过程中设置环境变量
ADD	(add): 将宿主机目录下的文件拷贝进镜像且 ADD 命令会自动处理 URL 和解压缩 tar 包
COPY	(copy): 类似 ADD,将宿主机目录下的文件拷贝文件和目录到镜像中 将从构建上下文目录中<源路径>的文件/目录复制到新的一层的镜像内的<目标路径>位置 COPY src dest 或 COPY ["src", "dest"]
VOLUME	(volume): 容器数据卷,用于数据保存和持久化工作
CMD	(cmd): 指定一个启动时要运行的命令 Dockerfile 中可以有多条 CMD 指令,但只有最后一个生效,CMD 会被 docker run 之后的参数覆盖
ENTRYPOINT	(entrypoint): 指定一个容器启动时要运行的命令 docker run 之后的参数会当做参数传递给 ENTRYPOINT 保留字指令(即追加),形成新的命令组合
ONBUILD	(onbulid): 当构建一个被继承的 Dockerfile 时运行命令,父镜像在被继承后父镜像的 onbuild 被触发

Docker-Dockerfile 自定义镜像:



自定义镜像示例一： Dockerfile 保留字指令

1. 编写: vi /MyDocker/Dockerfile

注: 注释内容不能和保留字指令在同一行

```
# FROM 继承本地的 centos 镜像
FROM centos
# 启动容器后,终端的所在目录
#ORKDIR /tmp
ENV mypath /tmp
WORKDIR $mypath
# vim 文本文件编辑
RUN yum -y install vim
# ifconfig 查看
RUN yum -y install net-tools
# 暴露出端口号
EXPOSE 80
CMD /bin/bash
```

2. 构建: docker build -f Dockerfile -t 新镜像名称:TAG 标签 .

如: `docker build -f /MyDocker/Dockerfile -t test/centos:1.1 .`

```
[root@CentosServer MyDocker]# docker build -f /MyDocker/Dockerfile -t test/centos:1.1 .
Sending build context to Docker daemon 2.048kB
Step 1/7 : FROM centos
--> 0f3e07c0138f
Step 2/7 : ENV mypath /tmp
--> Using cache
--> 32a19bd69e1b
Step 3/7 : WORKDIR $mypath
```

```
Removing intermediate container 2e7198a3b450
--> c2174ea6504d
Successfully built c2174ea6504d
Successfully tagged test/centos:1.1
[root@CentosServer MyDocker]#
```

3. 运行: `docker run -it` 新镜像名称:TAG 标签

如: `docker run -it test/centos:1.1`

```
[root@CentosServer MyDocker]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
test/centos         1.1                c2174ea6504d       3 hours ago        289MB

[root@CentosServer MyDocker]# docker run -it test/centos:1.1
[root@99f53c3c55e9 tmp]#
```

4. 列出镜像的变更历史: `docker history` 镜像名称/镜像 ID

如: `docker history test/centos:1.1`

```
[root@CentosServer MyDocker]# docker history test/centos:1.1
IMAGE               CREATED             CREATED BY          SIZE      COMMENT
c2174ea6504d        3 hours ago        /bin/sh -c #(nop)  CMD ["/bin/sh" "-c" "/bin...  0B
5f3c17d9c1fa        3 hours ago        /bin/sh -c #(nop)  EXPOSE 80                0B
e7581062dd78        3 hours ago        /bin/sh -c yum -y  install net-tools    14.3MB
a7669129f258        3 hours ago        /bin/sh -c yum -y  install vim          55.3MB
4b8aeab8ac6d        3 hours ago        /bin/sh -c #(nop)  WORKDIR /tmp          0B
32a19bd69e1b        3 hours ago        /bin/sh -c #(nop)  ENV mypath=/tmp          0B
0f3e07c0138f        4 weeks ago        /bin/sh -c #(nop)  CMD ["/bin/bash"]          0B
<missing>           4 weeks ago        /bin/sh -c #(nop)  LABEL org.label-schema.sc...  0B
<missing>           4 weeks ago        /bin/sh -c #(nop)  ADD file:d6fdacc1972df524a... 220MB
[root@CentosServer MyDocker]#
```

自定义镜像示例二: CMD 与 ENTRYPOINT

CMD 保留字指令: 覆盖

指定一个容器启动时要运行的命令,Dockerfile 中可以有多条 CMD 指令,但只有最后一个生效,CMD 会被 `docker run` 之后的参数替换,如下所示:

命令: `docker run -it -p 6666:8080 tomcat ls -l`

如图所示: `ls -l` 参数命令覆盖了 Dockerfile 中 CMD 指令: `catalina.sh run`

```
[root@CentosServer ~]# docker run -it -p 6666:8080 tomcat ls -l
total 132
-rw-r--r-- 1 root root 19318 Oct 7 13:33 BUILDING.txt
-rw-r--r-- 1 root root 5407 Oct 7 13:33 CONTRIBUTING.md
-rw-r--r-- 1 root root 57011 Oct 7 13:33 LICENSE
-rw-r--r-- 1 root root 1726 Oct 7 13:33 NOTICE
-rw-r--r-- 1 root root 3255 Oct 7 13:33 README.md
-rw-r--r-- 1 root root 7136 Oct 7 13:33 RELEASE-NOTES
-rw-r--r-- 1 root root 16262 Oct 7 13:33 RUNNING.txt
drwxr-xr-x 2 root root 4096 Oct 19 02:26 bin
drwxr-sr-x 2 root root 74 Oct 19 02:25 conf
drwxr-xr-x 2 root root 4096 Oct 19 02:25 lib
drwxrwxrwx 2 root root 6 Oct 7 13:30 logs
drwxr-sr-x 3 root staff 4096 Oct 19 02:25 native-jni-lib
drwxrwxrwx 2 root root 29 Oct 19 02:25 temp
drwxr-xr-x 7 root root 76 Oct 7 13:31 webapps
drwxrwxrwx 2 root root 6 Oct 7 13:30 work
[root@CentosServer ~]#
[root@CentosServer ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
e9af50124de6       tomcat             "catalina.sh run"   3 minutes ago       Up 3 minutes       8080/tcp           frosty_hellma
n
```

[root@CentosServer ~]# 显示结果表示容器并未被启动

编辑: vi dockerfile2

FROM centos

RUN yum -y install curl

CMD curl -s https://ip.cn

构建: docker build -f /MyDocker/dockerfile2 -t centosip .

```
Successfully built d479278e5a78
Successfully tagged centosip:latest
[root@CentosServer MyDocker]#
```

运行: docker run -it centosip

```
[root@CentosServer MyDocker]# docker run -it centosip3
{"ip": "114.247.186.84", "country": "北京市", "city": "联通"}
```

ENTRYPOINT 保留字指令: 追加

指定一个启动时要运行的命令,docker run 之后的参数会当做参数传递给 ENTRYPOINT 保留字指令(即追加),形成新的命令组合,如下所示:

编辑: vi dockerfile4

FROM centos

RUN yum -y install curl

ENTRYPOINT ["curl", "-s", "https://ip.cn"]

构建: docker build -f /MyDocker/dockerfile4 -t centosip4 .

```
Removing intermediate container 2be325e81080
--> 669283f3d162
Successfully built 669283f3d162
Successfully tagged centosip4:latest
[root@CentosServer MyDocker]#
```

运行: docker run -it centosip4

```
[root@CentosServer MyDocker]# docker run centosip6
{"ip": "114.247.186.84", "country": "北京市", "city": "联通"}
[root@CentosServer MyDocker]# docker run centosip6 -i
HTTP/2 200
Date: Wed, 30 Oct 2019 08:00:55 GMT
Content-Type: application/json; charset=UTF-8
Set-Cookie: __cfduid=de377708ef3bf27025c997f3aa5c060991572422455; expires=Thu, 29-Oct-20 08:00:55 GMT; path=/; domain=.ip.cn; HttpOnly
cf-cache-status: DYNAMIC
Expect-CT: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
Server: cloudflare
cf-ray: 52dbdbb849669911-LAX

{"ip": "114.247.186.84", "country": "北京市", "city": "联通"}
[root@CentosServer MyDocker]#
```

自定义镜像示例三： ONBUILD

保留字指令: ONBUILD

构建父镜像: MyCentOS_Father

编辑: vi dockerfile_father

```
FROM centos
RUN yum -y install curl
ENTRYPOINT ["curl", "-s", "https://ip.cn"]
ONBUILD RUN echo "father image onbuild ... 886"
```

构建: docker build -f /MyDocker/dockerfile_father -t MyCentOS_Father .

```
[root@CentosServer MyDocker]# vi dockerfile_father
[root@CentosServer MyDocker]# docker build -f /MyDocker/dockerfile_father -t mycentos_father .
Sending build context to Docker daemon 8.192kB
Step 1/4 : FROM centos
--> 0f3e07c0138f
Step 2/4 : RUN yum -y install curl
--> Using cache
--> c3ac93f5496b
Step 3/4 : ENTRYPOINT ["curl", "-s", "https://ip.cn"]
--> Running in 72151e9cef40
Removing intermediate container 72151e9cef40
--> 09f4ae578c14
Step 4/4 : ONBUILD RUN echo father image onbuild ... 886
--> Running in a868494bf88a
Removing intermediate container a868494bf88a
--> 2bb64d19d871
Successfully built 2bb64d19d871
Successfully tagged mycentos_father:latest
[root@CentosServer MyDocker]#
```

运行:

构建父镜像: MyCentOS_Son

编辑: vi dockerfile_son

```
FROM mycentos_father
```



```
RUN yum -y install curl
ENTRYPOINT ["curl","-s","https://ip.cn"]
```

构建: docker build -f /MyDocker/dockerfile_son -t mycentos_son .

```
[root@CentosServer MyDocker]# docker build -f /MyDocker/dockerfile_son -t mycentos_son .
Sending build context to Docker daemon 8.192kB
Step 1/3 : FROM mycentos_father
# Executing 1 build trigger
---> Running in 7d0020c70642
father image onbuild ... 886
Removing intermediate container 7d0020c70642
---> 26c043a85c29
Step 2/3 : RUN yum -y install curl
---> Running in 6eb6b9591f28
Last metadata expiration check: 1:17:52 ago on Wed Oct 30 07:32:45 2019.
Package curl-7.61.1-8.el8.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
Removing intermediate container 6eb6b9591f28
---> 179d16fc2edc
Step 3/3 : ENTRYPOINT ["curl","-s","https://ip.cn"]
---> Running in 5df09af758d9
Removing intermediate container 5df09af758d9
---> a254ca95e9e4
Successfully built a254ca95e9e4
Successfully tagged mycentos_son:latest
[root@CentosServer MyDocker]#
```

运行:

```
-rw-r--r--. 1 root root 125 10月 30 16:49 dockerfile_father
-rw-r--r--. 1 root root 88 10月 30 16:43 dockerfile_son
```

```
[root@CentosServer MyDocker]# docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
mycentos_son        latest          a254ca95e9e4   2 minutes ago  239MB
mycentos_father     latest         2bb64d19d871   3 minutes ago  238MB
```

自定义镜像示例四: ADD 与 COPY

构建自己的 Tomcat8 服务:

```
# 创建目录: mkdir -p /workspace/tomcat8
# 新建文件: touch /workspace/tomcat8/aaa.txt
# 将 jdk8 和 tomcat8 安装压缩包拷贝到该目录: 如下图示
```

```
[root@CentosServer tomcat8]# pwd
/workspace/tomcat8
[root@CentosServer tomcat8]# ll
总用量 197352
-rw-r--r--. 1 root root      0 10月 30 17:11 aaa.txt
-rw-r--r--. 1 root root 10267082 10月 30 17:04 apache-tomcat-8.5.47.tar.gz
-rw-r--r--. 1 root root 191817140 2月 27 2019 jdk-8u201-linux-x64.tar.gz
[root@CentosServer tomcat8]#
```

在/workspace/tomcat8/目录下新建 Dockerfile 文件并添加如下内容: vi Dockerfile


```

FROM centos
# 新镜像的维护者姓名
MAINTAINER chen
# 把宿主机当前目录下的 aaa.txt 拷贝到容器/usr/local/路径下并重新命名
COPY aaa.txt /usr/local/cincontainer.txt
# 把 java8 与 tomcat8 安装压缩包文件添加到容器中
ADD apache-tomcat-8.5.47.tar.gz /usr/local/
ADD jdk-8u201-linux-x64.tar.gz /usr/local/
# 安装 vim 编辑器
RUN yum -y install vim
# 设置工作访问时候的 WORKDIR 路径,登录落脚点
ENV MYPATH /usr/local
WORKDIR $MYPATH
# 配置 java8 与 tomcat8 环境变量
ENV JAVA_HOME /usr/local/jdk1.8.0_201
ENV CLASSPATH $JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
ENV CATALINA_HOME /usr/local/apache-tomcat-8.5.47
ENV CATALINA_BASE /usr/local/apache-tomcat-8.5.47
ENV PATH $PATH:$JAVA_HOME/bin:$CATALINA_HOME/lib:$CATALINA_HOME/bin
# 容器运行时监听的端口
EXPOSE 8080
# 启动时运行 tomcat
# ENTRYPOINT ["/usr/local/apache-tomcat-8.5.47/bin/startup.sh" ]
# CMD ["/usr/local/apache-tomcat-8.5.47/bin/catalina.sh","run"]
CMD          /usr/local/apache-tomcat-8.5.47/bin/startup.sh      &&          tail          -F
              /usr/local/apache-tomcat-8.5.47/bin/logs/catalina.out

```

```

[root@CentosServer tomcat8]# ll
总用量 197356
-rw-r--r--. 1 root root          0 10月 30 17:11 aaa.txt
-rw-r--r--. 1 root root 10267082 10月 30 17:04 apache-tomcat-8.5.47.tar.gz
-rw-r--r--. 1 root root      1094 10月 31 09:16 Dockerfile
-rw-r--r--. 1 root root 191817140 2月 27 2019 jdk-8u201-linux-x64.tar.gz

```

构建新镜像: **docker build -f /workspace/tomcat8/Dockerfile -t chen/tomcat8 .**

如果: Dockerfile 在当前目录下,可简写如: **docker build -t chen/tomcat8 .**

```

[root@CentosServer tomcat8]# docker build -t chen/tomcat8 .
Sending build context to Docker daemon 202.1MB
Step 1/15 : FROM centos
--> 0f3e07c0138f

```

```

Successfully built e95cf5fa6e5c
Successfully tagged chen/tomcat8:latest
[root@CentosServer tomcat8]#

```

运行新容器: 挂载宿主机目录运行容器

```
docker run -d -p 8888:8080 --name mytomcat8 -v /workspace/tomcat8/test:/usr/local/apache-tomcat-8.5.47/webapps/test -v /workspace/tomcat8/logs:/usr/local/apache-tomcat-8.5.47/logs/ --privileged=true chen/tomcat8
```

参数解析:

-d: 以守护进程的方式开启容器

-p: 8888:8080 (p 小写), 指定宿主机端口号与镜像的容器端口号

--name: 为开启的容器指定一个名字

-v: 容器数据卷, 宿主机的绝对路径目录: 容器内的绝对路径目录

即: test 是 tomcat 发布的项目, logs 目录下存放的 tomcat 运行项目的日志信息文件

Docker 挂载主机目录, Docker 访问出现 cannot open directory .: Permission denied

解决办法: 在挂载目录后多加一个 --privileged=true 参数即可

```
[root@CentosServer tomcat8]# docker images chen/tomcat8
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
chen/tomcat8         latest          e95cf5fa6e5c   17 minutes ago 686MB

[root@CentosServer tomcat8]# docker run -d -p 8888:8080 --name mytomcat8 -v /workspace/tomcat8/test:/usr/local/apache-tomcat-8.5.47/webapps/test -v /workspace/tomcat8/logs:/usr/local/apache-tomcat-8.5.47/logs/ --privileged=true chen/tomcat8
b2590cc2a8f2ee75face208e6c877080d32354c50d68bee9c728b0c244984707

[root@CentosServer tomcat8]# docker ps
CONTAINER ID        IMAGE             COMMAND                  CREATED            STATUS              PORTS               NAMES
b2590cc2a8f2        chen/tomcat8      "/bin/sh -c '/usr/lo..." 24 seconds ago    Up 20 seconds      0.0.0.0:8888->8080/tcp   mytomcat8
```

访问验证: <http://宿主机 IP:设置的端口> ---> 即: <http://localhost:8888>

注: 当容器是以守护进程的方式(-d)开启的, 要想访问容器内的目录或文件, 使用 exec 命令

如: docker exec b2590cc2a8f2 java -version

```
[root@CentosServer tomcat8]# docker exec b2590cc2a8f2 java -version
java version "1.8.0_201"
Java(TM) SE Runtime Environment (build 1.8.0_201-b09)
Java HotSpot(TM) 64-Bit Server VM (build 25.201-b09, mixed mode)
[root@CentosServer tomcat8]#
```

验证是否成功: 进入到 test 工程项目目录

宿主机: mkdir WEB-INF

vi web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
  id="WebApp_ID" version="2.5">

  <display-name>test</display-name>

</web-app>
```

vi a.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
  pageEncoding="UTF-8"%>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1">
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
<body>

---welcome docker tomcat8 test---
<%= "i am in docker tomcat self"%>
<br/>
<% System.out.println("---> docker tomcat self ."); %>

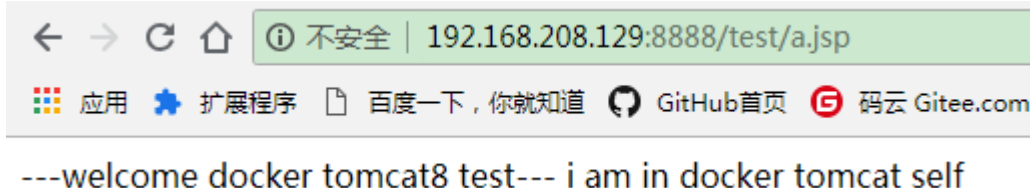
</body>
</html>
```

查看容器卷: `docker exec b2590cc2a8f2 ls -l /usr/local/apache-tomcat-8.5.47/webapps/test`

```
[root@CentosServer test]# docker exec b2590cc2a8f2 ls -l /usr/local/apache-tomcat-8.5.47/webapps/test
total 8
drwxr-xr-x. 2 root root 6 Oct 31 02:06 WEB-INF
-rw-r--r--. 1 root root 505 Oct 31 02:20 a.jsp
-rw-r--r--. 1 root root 332 Oct 31 02:09 web.xml
[root@CentosServer test]#
```

上图所示: 表示在宿主机中 `test` 目录下添加的文件共享到容器内的 `test` 目录下了
重启容器,查看 `web` 工程 `test` 发布情况

```
[root@CentosServer test]# docker restart b2590cc2a8f2
b2590cc2a8f2
[root@CentosServer test]#
```



---welcome docker tomcat8 test--- i am in docker tomcat self

查看日志信息: `cat /workspace/tomcat8/logs/catalina.out`

Docker 上传镜像到云服务器: 如 阿里云

登陆阿里云镜像容器服务:

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

新镜像的生成方式：两种方式

使用 Dockerfile 文件" `docker build -f Dockerfile -t 新镜像名称:TAG 标签` ."构建新的镜像文件

`docker commit -a="提交的镜像作者" -m="提交的镜像说明" 容器名称/容器 ID 新镜像名称:TAG 标签`

将新镜像推送到阿里云仓库:

阿里云镜像服务中创建仓库:

创建命名空间名称(即镜像前缀): `cd**521****`

创建镜像仓库名称(即镜像名称): 如 `tomcat8 mysql redis`

推送镜像到阿里云仓库:

`docker login --username=阿里云账号 registry.cn-shenzhen.aliyuncs.com`

`docker tag [ImageId] registry.cn-shenzhen.aliyuncs.com/cd**521****/tomcat:[镜像版本号]`

`docker push registry.cn-shenzhen.aliyuncs.com/cd**521****/tomcat:[镜像版本号]`

```
[root@CentosServer ~]# docker images chen/redis:3.2
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
chen/redis           3.2                1d84792a4f7e       48 minutes ago     76MB
[root@CentosServer ~]# docker login --username= registry.cn-shenzhen.aliyuncs.com
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[root@CentosServer ~]# docker tag 1d84792a4f7e registry.cn-shenzhen.aliyuncs.com/redis:3.2
[root@CentosServer ~]# docker push registry.cn-shenzhen.aliyuncs.com/redis:3.2
The push refers to repository [registry.cn-shenzhen.aliyuncs.com/redis]
094433da0a38: Pushed
56431c543d6c: Pushed
bb617143299d: Pushed
cfe17e3394d7: Pushed
a1a19279a9a: Pushed
197ffb073b01: Pushed
237472299760: Pushed
3.2: digest: sha256:373119721314a7fee9991aa34002a1972cb2a6d43f93d8224c8b912f3428895d size: 1778
```

将阿里云镜像拉取到本地:

#从阿里云仓库拉取镜像:

`docker pull registry.cn-shenzhen.aliyuncs.com/cd**521****/redis:[镜像版本号]`

```
[root@CentosServer ~]# docker pull registry.cn-shenzhen.aliyuncs.com/redis:3.2
3.2: Pulling from registry.cn-shenzhen.aliyuncs.com/redis
f17d81b4b692: Already exists
b32474098757: Already exists
8980cabe8bc2: Already exists
58af19693e78: Already exists
a977782cf22d: Already exists
9c1e268980b7: Already exists
2eb732a0f6aa: Pull complete
Digest: sha256:373119721314a7fee9991aa34002a1972cb2a6d43f93d8224c8b912f3428895d
Status: Downloaded newer image for registry.cn-shenzhen.aliyuncs.com/redis:3.2
registry.cn-shenzhen.aliyuncs.com/redis:3.2
```

CentOS7.2-Docker 常用安装服务:

安装 Tmcat8 镜像服务:

1. 创建宿主机与容器卷目录:

```
mkdir -p /workspace/cdtomcat/conf  
mkdir -p /workspace/cdtomcat/logs  
mkdir -p /workspace/cdtomcat/webapps
```

2. 准备 tomcat 与 jdk 压缩包:

```
-rw-r--r--. 1 root root 10267082 10月 31 11:14 apache-tomcat-8.5.47.tar.gz  
-rw-r--r--. 1 root root 940 10月 31 11:17 Dockerfile  
-rw-r--r--. 1 root root 191817140 10月 31 11:14 jdk-8u201-linux-x64.tar.gz
```

3. 编辑 Dockerfile 文件:

```
FROM centos  
# 新镜像的维护者姓名  
MAINTAINER chen<cdzm5211314@163.com>  
# 把 java8 与 tomcat8 安装压缩包文件添加到容器中  
ADD apache-tomcat-8.5.47.tar.gz /usr/local/  
ADD jdk-8u201-linux-x64.tar.gz /usr/local/  
# 设置工作访问时候的 WORKDIR 路径,登录落脚点  
ENV MYPATH /usr/local  
WORKDIR $MYPATH  
# 配置 java8 与 tomcat8 环境变量  
ENV JAVA_HOME /usr/local/jdk1.8.0_201  
ENV CLASSPATH $JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar  
ENV CATALINA_HOME /usr/local/apache-tomcat-8.5.47  
ENV CATALINA_BASE /usr/local/apache-tomcat-8.5.47  
ENV PATH $PATH:$JAVA_HOME/bin:$CATALINA_HOME/lib:$CATALINA_HOME/bin  
# 容器运行时监听的端口  
EXPOSE 8080
```

```
# 启动时运行 tomcat
# ENTRYPOINT ["/usr/local/apache-tomcat-8.5.47/bin/startup.sh" ]
# CMD ["/usr/local/apache-tomcat-8.5.47/bin/catalina.sh","run"]
CMD      /usr/local/apache-tomcat-8.5.47/bin/startup.sh      &&      tail      -F
/usr/local/apache-tomcat-8.5.47/bin/logs/catalina.out
```

4. 构建新的镜像文件:

命令: `docker build -f /workspace/cdtomcat/Dockerfile -t chen/tomcat8 .`

5. 运行新容器: 挂载宿主机目录与容器卷目录关系

命令如下:

```
docker      run      -d      -p      8080:8080      --name      mytomcat8      -v
/workspace/cdtomcat/webapps:/usr/local/apache-tomcat-8.5.47/webapps/      -v
/workspace/cdtomcat/logs:/usr/local/apache-tomcat-8.5.47/logs/      --privileged=true
chen/tomcat8
```

安装 MySQL5.7 镜像服务:

1. 拉取 5.7 版本 mysql 镜像:

命令: `docker pull mysql:5.7`

2. 创建宿主机与容器卷目录:

```
mkdir -p /workspace/cdmysql/conf
mkdir -p /workspace/cdmysql/logs
mkdir -p /workspace/cdmysql/data
```

3. 运行新容器:挂载宿主机目录与容器目录关系

```
docker run -d -p 3306:3306 --name mymysql5 -v /workspace/cdmysql/conf:/etc/mysql/conf.d -v /workspace/cdmysql/logs:/logs -v /workspace/cdmysql/data:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=root mysql:5.7
```

命令参数说明:

-d : 守护式进程开启容器(后台运行)

-p 3306:3306 :将宿主机的 3306 端口映射到 docker 容器的 3306 端口

--name mymysql5 :为运行的容器取个名称

-v /workspace/cdmysql/conf:/etc/mysql/conf.d :将主机/workspace/cdmysql/conf 目录下的 conf/my.cnf 挂载到容器的 /etc/mysql/conf.d

-v /workspace/cdmysql/logs:/logs :将主机/workspace/cdmysql/logs 目录挂载到容器的/logs 目录

-v /workspace/cdmysql/data:/var/lib/mysql :将主机/workspace/cdmysql/data 目录挂载到容器的/var/lib/mysql 目录

-e MYSQL_ROOT_PASSWORD=123456 :初始化 root 用户的密码

4. 进入容器交互终端操作 mysql:

命令: docker exec -it 容器名称/容器 ID /bin/bash

5. 宿主机上备份数据库:

命令: docker exec 容器名称/容器 ID sh -c 'exec mysqldump --all-databases -uroot -p"123456" ' > /workspace/cdmysql/all-databases.sql

6. 使用容器副本生成新镜像:

docker commit -a="提交的镜像作者" -m="提交的镜像说明" 容器名称/容器 ID 新镜像名称:TAG 标签

```
[root@CentosServer workspace]# docker commit -a="chen" -m="mysql5.7" 1b3caaaaf654 chen/mysql:5.7
sha256:8628ca12e4be634283c38717bd7f2c68b3a9719bbdf608b39f886c4f3a69c5a3
[root@CentosServer workspace]# docker images chen/mysql:5.7
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
chen/mysql           5.7                8628ca12e4be       23 seconds ago    437MB
[root@CentosServer workspace]#
```

安装 Redis3.2 镜像服务:

1. 拉取 3.2 版本 redis 镜像:

命令: `docker pull redis:3.2`

2. 创建宿主机与容器卷目录:

```
mkdir -p /workspace/cdredis/conf
mkdir -p /workspace/cdredis/data
mkdir -p /workspace/cdredis/logs
```

3. 运行新容器:挂载宿主机目录与容器目录关系

```
docker run -d -p 6379:6379 --name myredis -v /workspace/cdredis/conf:/usr/local/etc/redis/redis.conf -v /workspace/cdredis/data:/data redis:3.2 redis-server /usr/local/etc/redis/redis.conf --appendonly yes
```

4. 宿主机下创建 **redis.conf** 文件

宿主机/xxx/conf/redis.conf 目录下新建 redis.conf 文件
`vim /xxx/conf /redis.conf` 添加 redis 的配置文件信息内容

5. 测试 **redis-cli** 链接 **redis** 服务:

命令: `docker exec -it 容器名称/容器 ID redis-cli`

6. 测试 redis 持久化文件生成:

```
[root@CentosServer conf]# docker exec -it 5db3d5700639 redis-cli
127.0.0.1:6379> set key1 value1
OK
127.0.0.1:6379> SHUTDOWN
[root@CentosServer conf]# ls -l /workspace/cdredis/data/
总用量 4
-rw-r--r--. 1 systemd-bus-proxy ssh_keys 58 10月 31 15:24 appendonly.aof
[root@CentosServer conf]#
```

7. 使用容器副本生成新镜像:

`docker commit -a="提交的镜像作者" -m="提交的镜像说明" 容器名称/容器 ID 新镜像名称:TAG 标签`

```
[root@CentosServer workspace]# docker commit -a="chen" -m="redis3.2" 5db3d5700639 chen/redis:3.2
sha256:1d84792a4f7efa676a77e7eaca12aaa66f3a650f7db8c837f4ba2c1e3029c3ea
[root@CentosServer workspace]# docker images chen/redis:3.2
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
chen/redis           3.2                1d84792a4f7e       16 seconds ago     76MB
[root@CentosServer workspace]#
```

#####

CentOS7.2 宿主机: 创建目录

```
mkdir -p /workspace/cdtomcat/conf
mkdir -p /workspace/cdtomcat/web
mkdir -p /workspace/cdtomcat/logs
```

```
mkdir -p /workspace/cdmysql/conf
mkdir -p /workspace/cdmysql/data
mkdir -p /workspace/cdmysql/logs
```

```
mkdir -p /workspace/cdnginx/conf
mkdir -p /workspace/cdnginx/data
mkdir -p /workspace/cdnginx/logs
```

```
mkdir -p /workspace/cdredis/conf
mkdir -p /workspace/cdredis/data
mkdir -p /workspace/cdredis/logs
```

```
mkdir -p /workspace/cdmongodb/conf
mkdir -p /workspace/cdmongodb/data
```

```
mkdir -p /workspace/cdmongodb/logs
```