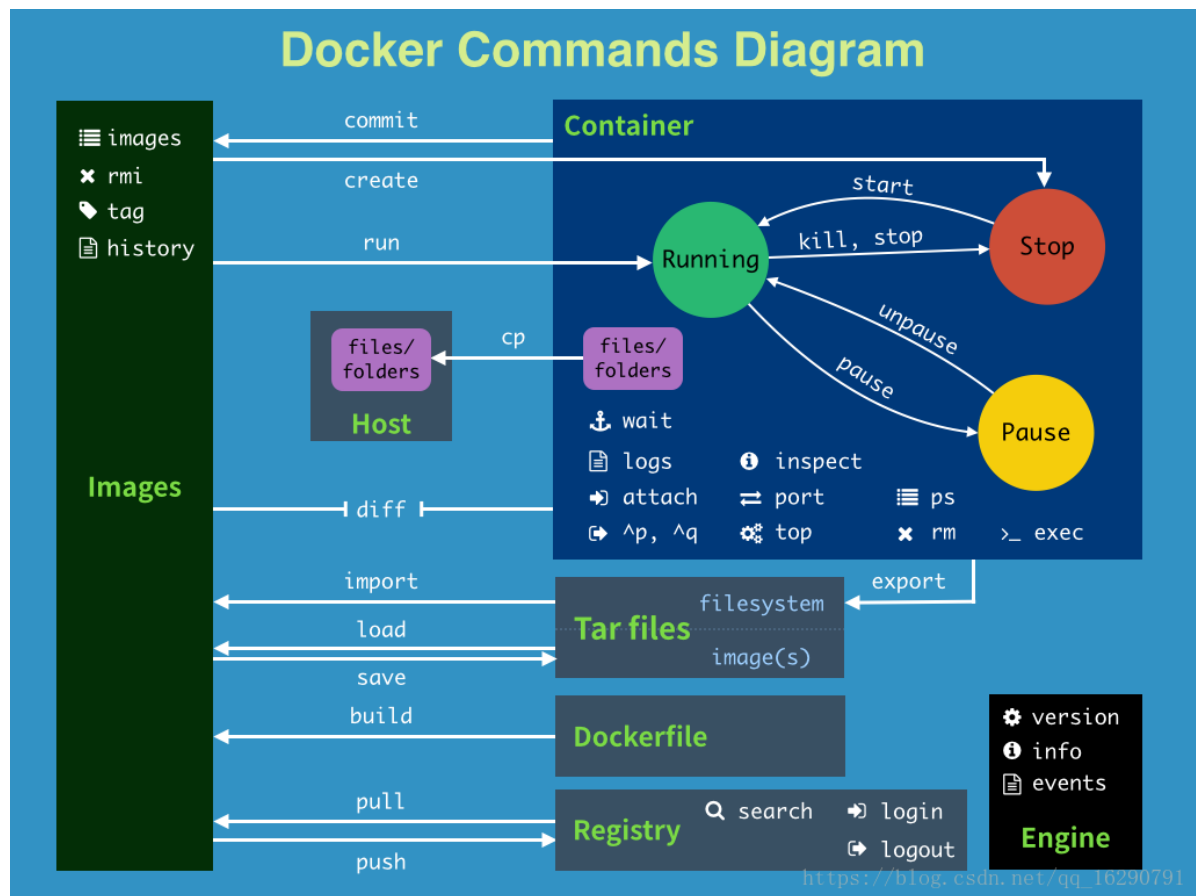


## docker命令图



## 帮助命令

### 查看docker版本 version

- docker version

### 查看docker信息，包括镜像和容器数量 info

- docker info

## 帮助命令 help

- docker [command] --help

# docker镜像命令

## 查看本地镜像 images

- docker images
  - -a -all # 列出所有镜像
  - -q --quiet # 只显示镜像id

## 搜索镜像 search

- docker search [ 镜像名 ]  
--filter=STARS=3000 # 列出收藏不少于指定值的镜像
- example  
docker search centos # 搜索centos镜像  
docker search --filter=STARS=3000 centos # 搜索收藏大于3000的centos镜像

## 下载镜像 pull

- docker pull [ 镜像名 ]:[ 版本号 ]  
-a, --all-tags # 下载所有镜像名的镜像
- docker pull nginx # 下载nginx镜像，默认为最新版
- docker pull nginx:1.14.0 # 下载版本为1.14.0的nginx镜像
- docker pull -a nginx # 下载仓库中名为nginx的所有镜像

## 删除镜像 rmi

- docker rmi [ 镜像名或者镜像id ]  
-f # 强制删除
- docker rmi -f 镜像id # 删除指定的镜像
- docker rmi -f 镜像id 镜像id 镜像id 镜像id # 删除指定的多个镜像
- docker rmi -f \$(docker images -aq) # 删除所有的镜像

# docker容器命令

## 启动容器 run

- docker run

-d # 后台运行容器，并返回容器ID  
-i # 以交互模式运行容器，通常与 -t 同时使用；  
-P # 随机端口映射，容器内部端口随机映射到主机的端口  
-p # 指定端口映射，格式为：主机(宿主)端口:容器端口  
-t # 为容器重新分配一个伪输入终端，通常与 -i 同时使用

### 常用

--name=" name " # 为容器指定一个名称  
--dns 8.8.8.8 # 指定容器使用的DNS服务器，默认和宿主一致  
--dns-search example.com # 指定容器DNS搜索域名，默认和宿主一致  
-h "hostname" # 指定容器的hostname  
-e username= "ritchie" # 设置环境变量  
--env-file=[] # 从指定文件读入环境变量  
--cpuset= "0-2" or --cpuset= "0,1,2" # 绑定容器到指定cpu运行  
-m # 设置容器使用内存最大值  
--net= "bridge" # 指定容器的网络连接类型，支持  
bridge/host/none/container：四种类型；  
--link=[] # 添加链接到另一个服务器  
--expose=[] # 开放一个端口或一组端口  
--volume -v # 绑定一个卷

- example

docker run -it centos /bin/bash # 启动并进入容器

docker run -it -v 主机目录:容器内目录 # 挂载目录

docker run -d 镜像名 # 问题：docker ps，发现容器停止了

常见问题，docker容器后台运行，就必须要有个前台进程，docker发现没有应用，会自动停止

## 查看容器信息 ps

- docker ps [CONTAINER ID or CONTAINER NAME]

-a # 列出当前正在运行的容器+历史运行过的容器  
-n=? # 显示最近创建的 ? 个容器  
-q # 只显示容器的ID

- **example**

```
docker ps          # 显示正在运行的容器
docker ps -n=3     # 显示最近三个创建的容器
```

## 退出容器 exit

- exit # 直接停止容器并退出
- Ctrl + P + Q # 容器不停止退出

## 删除容器 rm

- docker rm 容器id # 删除指定容器，不能删除正在运行的容器
- docker rm -f \$(docker ps -aq) # 删除全部容器
- docker ps -a -q | xargs docker rm # 删除全部容器

## 启动和停止容器的操作

- docker start 容器id # 启动容器
- docker restart 容器id # 重启容器
- docker stop 容器id # 停止当前正在运行的容器
- docker kill 容器id # 强制停止当前的容器

## 查看容器日志 logs

- docker logs -tf --tail 10 容器id # 查看容器的最新十条日志
  - t # 时间戳
  - f # 跟踪日志输出
  - tail # 列出多少条

## 查看容器中的进程信息 top

- docker top [ 容器id ] # 显示容器的进程信息

## 查看镜像的元数据 inspect

- docker inspect [ 容器id ]
  - f # 指定返回值的模板文件
- **example**

```
[root@iz2ze63uxgt90ev530ow6oz ~]# docker inspect -f
'{{.Config.Env}}' 1935a9975ef3 #获取容器环境变量
[PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin]
```

## 进入容器 exec attach

- `docker exec -it 容器id /bin/bash` # 进入容器后开启一个新终端
- `docker attach 容器id` # 进入容器后会进入正在执行的终端，不会开启新终端

## 文件拷贝 cp

- `docker cp 容器id : 容器内路径 目的主机路径`
- **example**

```
1. docker cp 8d2ab3694825:/home/test.java /home/ # 容器复制到宿主机
2. docker cp /home/test.java 8d2ab3694825:/home/ # 宿主机复制到容器
```

## 提交为一个新的镜像 commit

- `docker commit -m "提交的描述信息" -a "作者" 容器id 目标镜像名:版本号`

## 挂载卷 -v

- **指定路径挂载**
- `docker run -it -v 主机文件地址:仓库文件地址 仓库id /bin/bash`
- **example 挂载mysql**

```
[root@iz2ze63uxgt90ev530ow6oz ~]# docker run -d -p
3308:3306 -v /home/mysql/conf:/etc/mysql/conf.d -v
/home/mysql/data:/var/lib/mysql -e
MYSQL_ROOT_PASSWORD=123123 --name mysql1 mysql
```

- **匿名挂载、具名挂载、指定路径挂载区分**
- `-v 容器内路径` # 匿名挂载
- `-v 卷名:容器内路径` # 具名挂载

所有docker容器内的卷，没有指定目录的情况下都是在/var/lib/docker/volumes/xxx/\_data

- `-v /宿主机路径:容器内路径` # 指定路径挂载

- **example**

通过ro rw改变读写权限

ro     readonly     **只读**  
rw     readwrite    **可读可写**

```
[root@iz2ze63uxgt90ev530ow6oz ~]# docker run -d -P -v
/etc/nginx nginx --name nginx1
d2caee7eaae8e12e8e523effaea004e3026f77a17e32a218cccb06e617
46e510            # 匿名挂载
[root@iz2ze63uxgt90ev530ow6oz ~]# docker run -d -P -v
juming-nginx:/etc/nginx:rw nginx --name nginx2
3c24e5d740b4b3e84c695ac0af23571e6f2d74ca03db9575ac9dff9668
cb9202            # 具名挂载
[root@iz2ze63uxgt90ev530ow6oz ~]# docker volume ls
DRIVER                      VOLUME NAME
local
5de45d7f844d383adb164e191917788bee3e1393df5776543b54a544e4
be7a6f
local                      juming-nginx
[root@iz2ze63uxgt90ev530ow6oz ~]# docker inspect juming-
nginx
[
  {
    "CreatedAt": "2021-03-03T18:22:24+08:00",
    "Driver": "local",
    "Labels": null,
    "Mountpoint": "/var/lib/docker/volumes/juming-
nginx/_data",            #挂载路径
    "Name": "juming-nginx",
    "Options": null,
    "Scope": "local"
  }
]
```

只要看到ro就说明这个路径只能通过宿主机来操作，容器内无法操作

## Dockerfile

### 构建镜像 build

- docker build -f /home/docker-test-volume/dockerfile -t layman/dockerfile .  
-f        **# 指定要使用的Dockerfile路径**  
-t        **# 镜像的名字及标签**

# 制作tomcat镜像

文件结构

```
[root@iz2ze63uxgt90ev530ow6oz home]# tree build/
build/
├── tomcat
│   ├── apache-tomcat-9.0.43.tar.gz
│   ├── Dockerfile
│   ├── jdk-8u271-linux-x64.tar.gz
│   ├── readme.txt
│   ├── test
│   │   ├── index.jsp
│   │   └── WEB_INF
│   │       └── web.xml
│   └── tomcat-logs
│       ├── catalina.2021-03-07.log
│       ├── catalina.out
│       ├── host-manager.2021-03-07.log
│       ├── localhost.2021-03-07.log
│       ├── localhost_access_log.2021-03-07.txt
│       └── manager.2021-03-07.log
```

压缩包

发布文件夹、文件外部挂载

日志

*#Dockerfile*

FROM centos

MAINTAINER wll<1415155099@qq.com>

COPY readme.txt /usr/local/readme.txt

ADD jdk-8u271-linux-x64.tar.gz /usr/local/

ADD apache-tomcat-9.0.43.tar.gz /usr/local/

RUN yum -y install vim

ENV MYPATH /usr/local

WORKDIR \$MYPATH

ENV JAVA\_HOME /usr/local/jdk1.8.0\_271

ENV CLASSPATH \$JAVA\_HOME/lib/dt.jar:\$JAVA\_HOME/lib/tools.jar

ENV CATALINA\_HOME /usr/local/apache-tomcat-9.0.43

ENV CATALINA\_BASH /usr/local/apache-tomcat-9.0.43

ENV PATH

\$PATH:\$JAVA\_HOME/bin:\$CATALINA\_HOME/lib:\$CATALINA\_HOME/bin

EXPOSE 8080

CMD /usr/local/apache-tomcat-9.0.43/bin/startup.sh && tail -F /usr/local/apache-tomcat-9.0.43/bin/logs/catalina.out

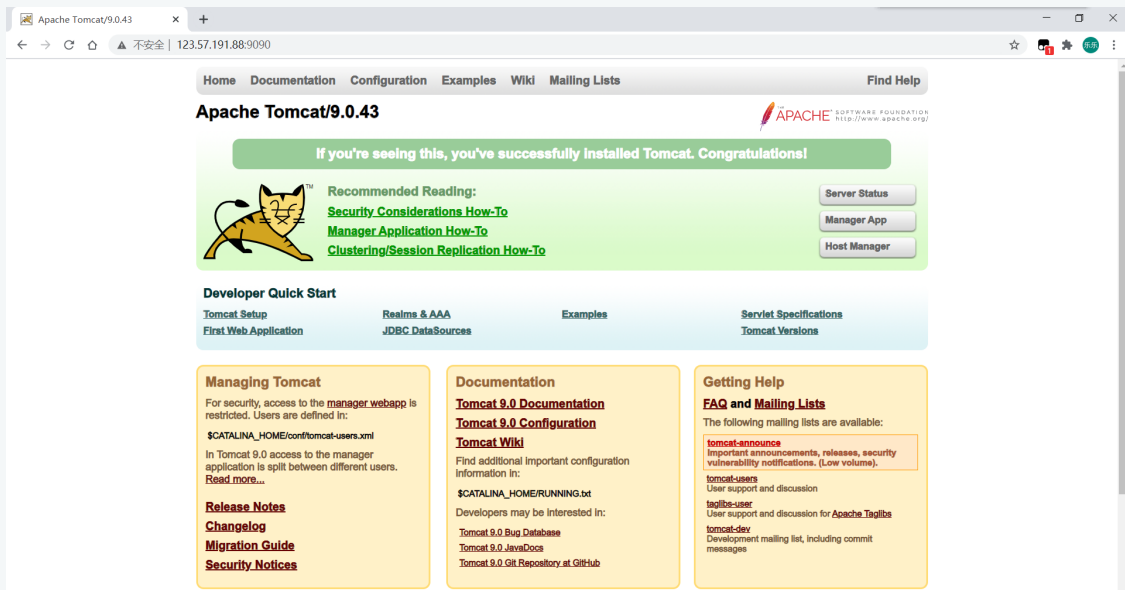
## 构建镜像

```
docker build -t diytomcat .
```

## 启动镜像

```
docker run -d -p 9090:8080 --name wll-tomcat -v  
/home/build/tomcat/test:/usr/local/apache-tomcat-  
9.0.43/webapps/test -v /home/build/tomcat/tomcat-  
logs:/usr/local/apache-tomcat-9.0.43/logs diy_tomcat:1.0
```

## 访问测试



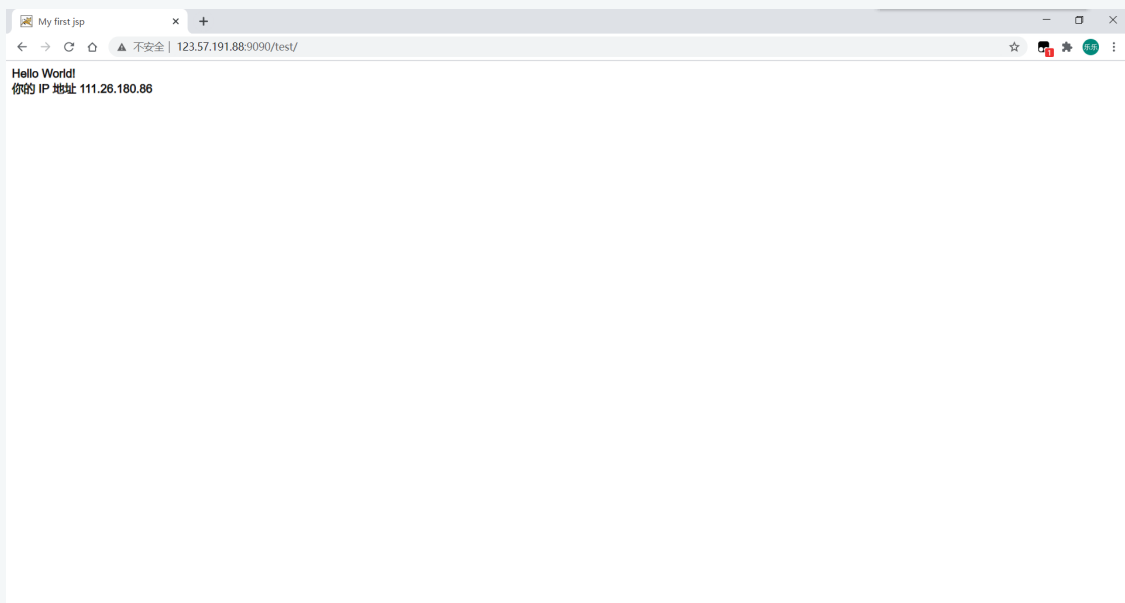
## 发布项目

由于做了卷挂载，所以直接在本地进行项目发布即可。

```
[root@iz2ze63uxgt90ev530ow6oz test]# cat index.jsp  
<%@ page language="java" contentType="text/html; charset=UTF-  
8"  
    pageEncoding="UTF-8"%>  
<!DOCTYPE html>  
<html>  
<head>  
<meta charset="utf-8">  
<title>My first jsp</title>  
</head>  
<body>  
Hello World!<br/>  
<%  
out.println("你的 IP 地址 " + request.getRemoteAddr());  
>%  
<%  
System.out.println("----my test web logs----");
```



```
%>
</body>
</html>
```



查看日志

```
[root@iz2ze63uxgt90ev530ow6oz tomcat-logs]# cat catalina.out
07-Mar-2021 04:09:13.078 INFO [main]
org.apache.coyote.AbstractProtocol.start Starting
ProtocolHandler ["http-nio-8080"]
07-Mar-2021 04:09:13.161 INFO [main]
org.apache.catalina.startup.Catalina.start Server startup in
[175726] milliseconds
----my test web logs----
----my test web logs----
----my test web logs----
```

## docker网络