# Docker

## 基础知识

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。

镜像

容器

仓库

### 容器数据卷

把容器内的数据持久化

本机目录映射关联容器目录（共享文件夹，双向操作均有效）

docker run -it -v /宿主机绝对路径目录 : /容器内目录 镜像名

docker run -it -v /myDataVolume : /dataVolumeContainer centos:6.8

绑定过后容器关闭了再启动容器该目录仍然会映射到本机目录，该目录数据则达到本机与容器同步的效果

权限控制

容器只能读取，不能写入

docker run -it -v /宿主机绝对路径目录 : /容器内目录:ro 镜像名

docker run -it -v /myDataVolume : /dataVolumeContainer:ro centos:6.8

## 安装

#### centos6.8

1.yum install -y epel-release

2.yum install -y docker-io

yum install <https://get.docker.com/rpm/1.7.1/centos-6/RPMS/x86_64/docker-engine-1.7.1-1.el6.x86_64.rpm>

3.安装后的配置文件：/etc/sysconfig/docker

4.启动Docker后台服务: service docker start

5.docker version验证是否安装成功

测试

https://dev.aliyun.com/search.html

运行测试镜像

docker run hello-world

#### centos7

## 命令

### 帮助命令

docker version 查看版本号

docker info 详细信息

docker --help 帮助

### 镜像命令

查看已有镜像

docker images

-a：列出本地所有的镜像（含中间映像层）

-q：只显示镜像D。

--digests：显示镜像的摘要信息

--no-trunc：显示完整的镜像信息

查询镜像名称

docker search [options] 镜像名

-s 点赞数排序 docker search -s 30 tomcat

--automated 只列出automated build类型的镜像；

拉取镜像

docker pull 镜像名

docker pull tomcat等价于docker pull tomcat:latest docker pull tomcat:3.2

删除镜像

docker rmi 镜像名…

docker rmi hello-wolrd

-f 强制删除 docker rmi -f hello-world

全部删除 docker rmi -f $(docker images -qa)

### 容器命令

新建并启动容器

docker run [OPTIONS] IMAGE[COMMAND][ARG..]

--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；

-i：以交互模式运行容器，通常与-t同时使用；

-t：为容器重新分配一个伪输入终端，通常与-i同时使用；

-P：随机端口映射；

-p：指定端口映射，有以下四种格式

ip:hostPort:containerPort

ip::containerPort

hostPort:containerPort

containerPort

-v 指定挂载目录

启动失败原因

1、可能语法错误（使用docekr create）

2、挂载的目录没有写入权限 chmod -R 777 /ghaya/mysql/data

3、磁盘满了（overlay文件系统满了，一般的镜像太多导致）

启动tomcat 容器端口8888：对应容器内tomcat端口8080

docker run -it -p 8888:8080 tomcat

动态分配端口 docker ps 可以查看

docker run -it -P tomcat

列出当前所有正在运行的容器

docker ps

-a：列出当前所有正在运行的容器+历史上运行过的

-i：显示最近创建的容器。

-n：显示最近n个创建的容器。 docker ps -n 10

-q：静默模式，只显示容器编号。

--no-trunc：不截断输出。

退出容器

exit 容器停止退出 / ctrl + D

ctrl+P+Q 容器不停止退出

启动容器

docker start 容器ID或者容器名

重启容器

docker restart 容器ID或者容器名

停止容器

docker stop 容器ID或者容器名

强制停止容器

docker kill 容器ID或者容器名

删除已停止的容器

docker rm 容器ID或者容器名

-f 强制

删多个 docker rm -f $(docker ps -aq)

docker ps -a -q |xargs docker rm

【重要start】

启动守护式容器

docker run -d 容器名

防止守护进程自杀

docker run -d centos:6.8 /bin/sh -c "while true;do echo hello zzyy;sleep 2;done"

查看容器日志

docker logs -f -t --tail 容器id

查看容器内运行的进程

docker top 容器id

查看容器内部细节

docker inspect 容器id

进入正在运行的容器并以命令行交互

1. 重新进入

docker attach 容器ID

1. 不进入容器直接执行命令

docker exec -it 容器lD bashshell docker exec -t 20cc171dcb99 ls -l /tmp

1. 进入容器

docker exec -it [容器id] bash

上述两个区别

attach直接进入容器启动命令的终端，不会启动新的进程

exec是在容器中打开新的终端，并且可以启动新的进程

从容器内拷贝文件到主机上

docker cp

docker cp 20cc171dcb99:/tmp/yum.log /tmp/yunFromDockerCentos68.log

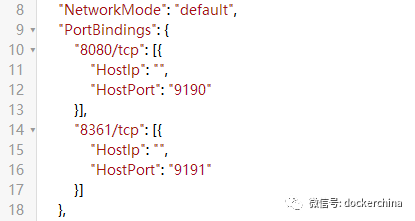
修改容器映射端口

容器的配置文件路径：

cd /var/lib/docker/containers/

/var/lib/docker/containers/[hash\_of\_the\_container]/hostconfig.json

/var/lib/docker/containers/83b4631150abefcda9158587f09118cc541fa665070ba2753eb0bcd9f15f7728/hostconfig.json



文件中其中有一项是PortBindings，其中8080/tcp对应的是容器内部的8080端口，HostPort对应的是映射到宿主机的端口9190。8361/tcp对应的是容器内部的8361端口，HostPort对应的是映射到宿主机的端口9191。按需修改端口，然后重启docker服务，再启动容器服务就可以了。

systemctl restart docker

【重要end】

### 生成镜像

将已有容器生成新的镜像，提交。

docker commit [options] [cid]

-m 提交信息

-a 作者

docker commit -a="ghaya" -m=" custom tomcat8.5" 509cbe53e226 ghaya/tomcat:latest

docker commit -a="ghaya" -m=" custom centos6.8" 0f0c6ed5715d ghaya/centos:latest

通过dockerfile编写建立新的镜像

构建dockerfile

docker build [options] -t [镜像名]:TAG .

-t 生成名:版本

-f 路径

列出镜像的变更历史

docker history 镜像名

## dockerFile

### Dockerile内容基础知识

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一

2：指令按照从上到下，顺序执行

3：#表示注释

4：每条指令都会创建一个新的镜像层，并对镜像进行提交

### Docker执行Dockerfile的大致流程—

（1）docker从基础镜像运行一个容器

（2）执行一条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中的下一条指令直到所有指令都执行完成

### 保留字指令

FROM 基础镜像，当前新镜像是基于哪个镜像的

MAINTAINER 镜像维护者的姓名和邮箱地址

RUN 容器构建时需要运行的命令

EXPOSE 当前容器对外暴露出的端口

WORKDIR 指定在创建容器后，终端默认登陆的进来工作目录，一个落脚点

ENV 用来在构建镜像过程中设置环境变量

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tar压缩包

COPY 类似ADD，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

VOLUME 容器数据卷，用于数据保存和持久化工作

和运行时 -v 关联映射挂载点作用一样

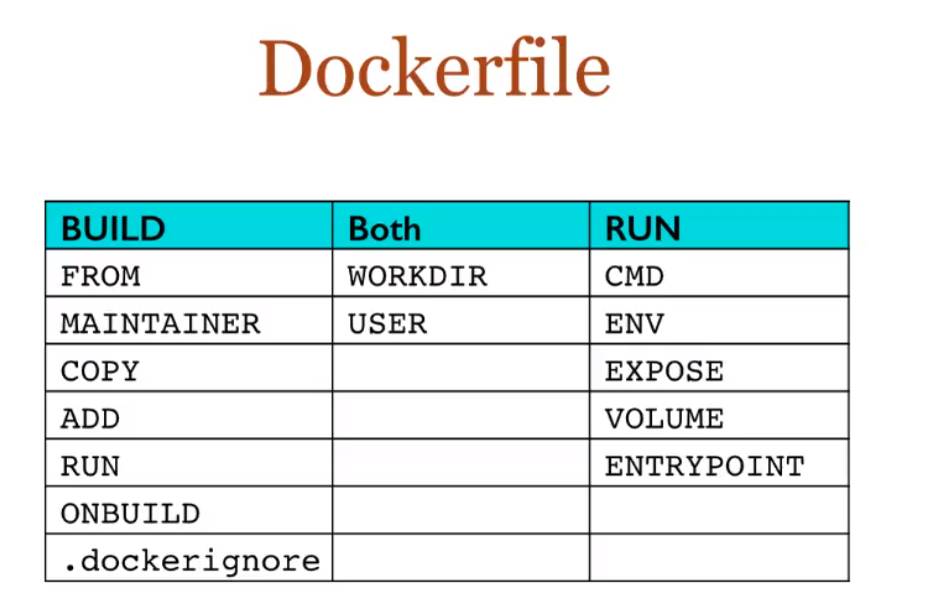
CMD 指定一个容器启动时要运行的命令团个

Dockerfile中可以有多个CMD指令，但只有最后一个生效，CMD会被dockerrun之后的参数替换（覆盖）

ENTRYPOINT 指定一个容器启动时要运行的命令

ENTRYPOINT的目的和CMD一样，都是在指定容器启动程序及参数(追加)

ONBUILD 当构建一个被继承的Dockerfile时运行命令，父镜像在被子继承后父镜像的pnbuild被触发



### 案例（重要）

#### 1 自定义mycentos目的使我们自己的镜像具备如下：

1.登陆后的默认路径

2.vim编辑器

3.查看网络配置ifconfig支持

**编写文件**

vim /ghaya/mydocker/dockerfile\_ghaya\_centos

#来自某个镜像

from ghaya/centos

#作者

MAINTAINER ghaya

#变量

ENV mypath /tmp

#登陆路径

WORKDIR $mypath

#启动

RUN yum -y install vim

#本镜像已经支持

#RUN yum -y install net-tools

#暴露端口

EXPOSE 80

CMD echo $mypath

CMD echo "ghaya centos start success --------- ok"

#进入命令行

CMD /bin/bash

**构建dockerfile**

**在当前目录可以不用-f**

docker build -f /ghaya/mydocker/dockerfile\_ghaya\_centos -t dfcentos:1.3 .



#### 2 ENTRYPOINT

FROM ghaya/centos

#安装 curl

RUN yum install -y curl

#用CMD可能会被覆盖

#CMD {"curl","-s","http://ip.cn"}

#追加命令

ENTRYPOINT {"curl","-s","http://ip.cn"}

docker build -f /ghaya/mydocker/dockerfile\_ghaya\_centosIP -t dfcentosIP .

#### 3 ONBUILD 被继承时触发

ONBUILD RUN echo " bei ji cheng"

#### 4 ADD、COPY

FROM centos

MAINTAINER zzyy<zzyybs@126.com>

#把宿主机当前上下文的c.txt拷贝到容器/usr/1ocal/路径下

COPY c.txt /usr/local/cincontainer.txt

#把java与tomcat添加到容器中

ADD jdk-8u171-linux-x64.tar.gz /usr/local/

ADD apache-tomcat-9.0.8.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java与tomcat环境变量

ENV JAVA HOME/usr/local/jdk1.8.0 171

ENV CLASSPATH $JAVA\_HOME/1ib/dt.jar：$JAVA HOME/1ib/tools.jar

ENV CATALINA HOME/usr/1ocal/apache-tomcat-9.0.8

ENV CATALINA BASE/usr/local/apache-tomcat-9.0.8

ENV PATH $PATH：$JAVA\_HOME/bin：$CATALINA\_HOME/1ib：$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-9.0.8/bin/startup.sh"]

#CMD ["/usr/1ocal/apache-tomcat-9.0.8/bin/catalina.sh","run"]

CMD /usr/local/apache-tomcat-9.0.8/bin/startup.sh && tail -F \

/usr/local/apache-tomcat-9.0.8/bin/logs/catalina.out

运行tomcat

docker run -d -p 9080:8080 --name myt9

-v /zzyyuse/mydockerfile/tomcat9/test:/usr/local/apache-tomcat-9.0.8/webapps/test

-v /zzyyuse/mydockerfile/tomcat9/tomcat9logs/:/usr/local/apache-tomcat-9.0.8/1ogs

--privileged=true zzyytomcat9



### 其他

#### 安装nginx

创建映射目录

mkdir -p /ghaya/nginx/www

mkdir -p /ghaya/nginx/logs

mkdir -p /ghaya/nginx/conf

#拷贝配置文件

docker cp [容器id]:/etc/nginx/nginx.conf /ghaya/nginx/conf

docker cp 8e1a3419c137:/etc/nginx/nginx.conf /ghaya/nginx/conf

#映射容器目录

docker run -d -p 80:80 --name nginx \

-v /ghaya/nginx/www:/usr/share/nginx/html \

-v /ghaya/nginx/conf/nginx.conf:/etc/nginx/nginx.conf \

-v /ghaya/nginx/logs:/var/log/nginx \

nginx

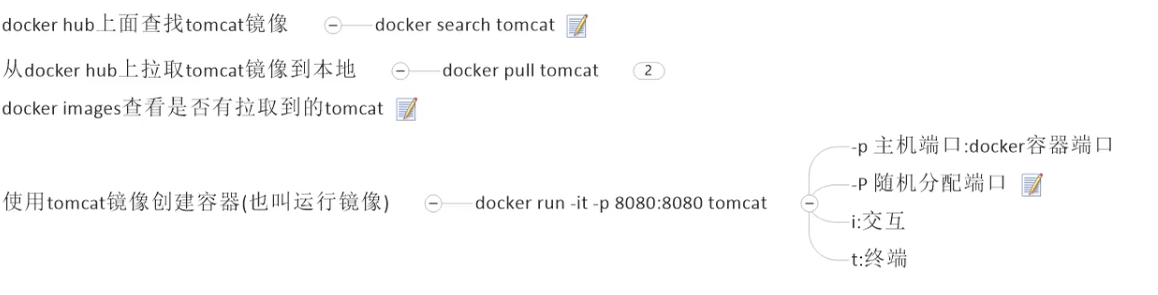
创建html

配置nginx

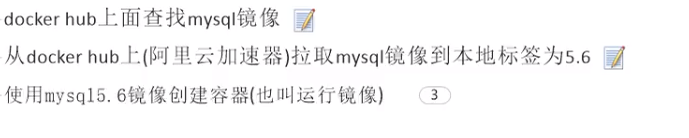




#### 安装tomcat



#### 安装mysql



拉取镜像

docekr pull mysql:5.7

创建容器 mysql:5.7

docker run \

-p 3307:3306 \

--name mysql \

-v /ghaya/mysql/conf:/etc/mysql/conf.d \

-v /ghaya/mysql/logs:/logs \

-v /ghaya/mysql/data:/var/1ib/mysql \

-e MYSQL\_ROOT\_PASSWORD=123456 \

-d ghaya/mysql

docker run -d \

-p 3308:3306 \

--name mysql8 \

--privileged=true \

-v /ghaya/mysql8/conf:/etc/mysql/conf.d \

-v /ghaya/mysql8/logs:/logs \

-v /ghaya/mysql8/data:/var/lib/mysql \

-v /etc/localtime:/etc/localtime \

-e MYSQL\_ROOT\_PASSWORD=123456 \

mysql:latest

docker run -d \

--name mysql8 \

-p 3308:3306 \

-e MYSQL\_ROOT\_PASSWORD=123456 \

mysql:8.0

docker run -it \

-p 3308:3306 \

--name mysql8 \

-e MYSQL\_ROOT\_PASSWORD=123456 \

mysql:8.0

命令说明：

#将主机的12345端口映射到docker容器的3306端口

-p 12345：3306

#运行服务名字

-name mysql

#将主机/zzyyuse/mysql录下的conf/my.cnf挂载到容器的/etc/mysql/conf.d

-v /zzyyuse/mysql/conf:/etc/mysql/conf.d

#将主机/zzyyuse/mysql目录下的logs目录挂载到容器的/logs

-v /zzyyuse/mysql/logs:/logs

#将主机/zzyyuse/mysql目录下的data目录挂载到容器的/var/lib/mysql

-v/zzyyuse/mysql/data:/var/lib/mysql

#初始化root用户的密码。

-e MYSQL\_ROOT\_PASSWORD=123456

#后台程序运行mysql5.6I

-d mysql:5.6

交互

启动之后

进入mysql命令窗口

docker exec -it mysql /bin/bash

登陆

mysql -uroot -p

123456

查看所有数据库

show databses;

测试数据

CREATE DATABASE db1 DEFAULT CHARSET utf8;

USE db1;

CREATE TABLE a1(id int);

insert into a1() values(1),(2);

CREATE TABLE a2(id int);

insert into a2() values(2);

CREATE TABLE a3(id int);

insert into a3() values(3);

CREATE DATABASE db2 DEFAULT CHARSET utf8;

USE db2;

CREATE TABLE b1(id int);

insert into b1() values(1);

CREATE TABLE b2(id int);

insert into b2() values(2);

容器id或者名字

数据备份

docker exec [myql容器ID] sh -c ' exec mysqldump --all-databases \

-uroot -p"123456" ' > /ghaya/all-databases.sql

docker exec mysql sh -c ' exec mysqldump --all-databases -uroot -p"123456" ' > /ghaya/mysql/all-databases20201212.sql

恢复

cat /ghaya/mysql/all-databases.sql | docker exec -i mysql /usr/bin/mysql -u root -p123456

cp慢查询日志

docker cp 83b4631150ab:/var/lib/mysql/83b4631150ab-slow.log /ghaya/mysql/logs\_other

安装问题

**caching\_sha2\_password**

**连接MySQL数据库出现时Authentication plugin 'caching\_sha2\_password' cannot be loaded的解决办法**

问题描述：用Navicat Premium或HeidiSQL连接MySQL数据库时会弹出下面的情况

**解决方法：**

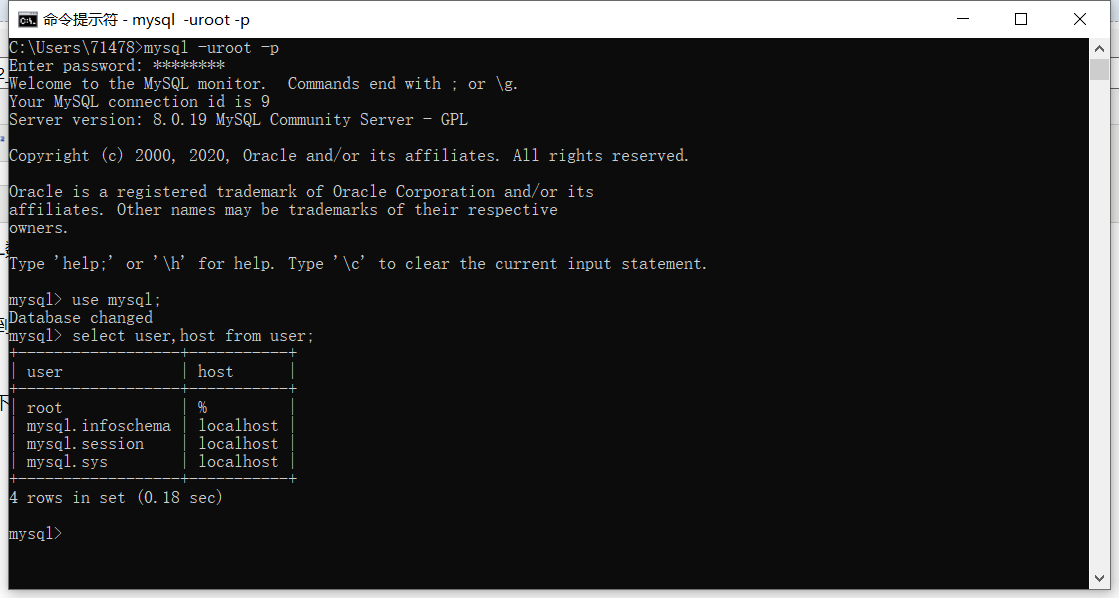
1、运行命令行窗口，输入以下命令，输入密码后进入到mysql中，(最好将MySQL安装目录的Bin文件夹路径加入到环境变量，这样就不用进入到Bin文件夹下了)

mysql -uroot -p

2、查看用户和主机信息，输入以下命令，显示界面如下

use mysql;

select user,host from user;



上面显示用户为root，host为%，说明是已经设置好远程连接了如果没有设置，

host为localhost

3、运行以下命令

ALTER USER 'root'@'%' IDENTIFIED BY '123456' PASSWORD EXPIRE NEVER;

ALTER USER 'root'@'%' IDENTIFIED WITH mysql\_native\_password BY '123456';

FLUSH PRIVILEGES；

4、再次连接，成功连接数据库

#### 安装redis

<https://www.jianshu.com/p/4413f484789d>

**一、找到一个合适的docker的redis的版本**

**二、使用docker安装redis**

sudo docker pull redis

**三、准备redis的配置文件**

因为需要redis的配置文件，这里最好还是去redis的官方去下载一个redis使用里面的配置文件即可

redis中文官方网站：[http://www.redis.cn/download.html](https://links.jianshu.com/go?to=http%3A%2F%2Fwww.redis.cn%2Fdownload.html" \t "_blank)

**四、配置redis.conf配置文件**

修改redis.conf配置文件：  
主要配置的如下：

bind 127.0.0.1 #注释掉这部分，使redis可以外部访问

protected-mode no #关闭保护模式

daemonize no #用守护线程的方式启动

requirepass 你的密码 #给redis设置密码

appendonly yes#redis 持久化　　默认是no

tcp-keepalive 300 #防止出现远程主机强迫关闭了一个现有的连接的错误

默认是300

**五、创建本地与docker映射的目录，即本地存放的位置**

创建文件夹

mkdir /data/redis

mkdir /data/redis/data

复制

cp -p redis.conf /data/redis/

**六、启动docker redis**

docker run -p 6379:6379 --name redis \

-v /ghaya/redis/redis.conf:/etc/redis/redis.conf \

-v /ghaya/redis/data:/data \

-d redis redis-server /etc/redis/redis.conf --appendonly yes

**七、查看是否启动成功**

#### 安装rabbitMQ

user: admin

pw: admin

<https://www.cnblogs.com/yy-cola/p/11089800.html>

1.下载镜像（有时候网络问题超时，多尝试几次即可。我这里选择的是可以访问web管理界面的tag）

docker pull rabbitmq:management

2.创建容器并运行（15672是管理界面的端口，5672是服务的端口。这里顺便将管理系统的用户名和密码设置为admin admin）

docker run -dit --name rabbitmq \

-e RABBITMQ\_DEFAULT\_USER=admin \

-e RABBITMQ\_DEFAULT\_PASS=admin \

-p 15672:15672 \

-p 5672:5672 rabbitmq:management

安装es

#### 安装ElasticSearch

安装

docker pull elasticsearch:7.4.2 本体

docker pull kibana:7.4.2 可视化工具

配置es

mkdir -p /ghaya/elasticsearch/config

mkdir -p /ghaya/elasticsearch/data

mkdir -p /ghaya/elasticsearch/plugins

chmod -R 777 /ghaya/elasticsearch

echo "http.host:0.0.0.0">>/mydata/elasticsearch/config/elasticsearchyml

挂载并启动es

docker run --name elasticsearch -p 9200:9200 -p 9300:9300 \

-e "discovery.type=single-node" \

-e ES\_JAVA\_OPTS="-Xms64m -Xmx128m" \

-v /ghaya/elasticsearch/config/elasticsearch.yml:/usr/share/elasticsearch/config/elasticsearch.yml \

-v /ghaya/elasticsearch/data:/usr/share/elasticsearch/data \

-v /ghaya/elasticsearch/plugins:/usr/share/elasticsearch/plugins \

-d elasticsearch:7.4.2

启动kibana可视化界面

linux上配不上去ip有问题 装了windows的

docker run --name kibana -e ELASTICSEARCH\_HOSTS=http://192.168.1.206:9200 -p 5601:5601 \

-d kibana:7.4.2

docker run -d --restart=always --log-driver json-file --log-opt max-size=100m --log-opt max-file=2 --name kibana -p 5601:5601 \

-v /ghaya/kibana/kibana.yml:/usr/share/kibana/config/kibana.yml kibana:7.4.2

docker run -it -d -e ELASTICSEARCH\_URL=http://192.168.1.206.1:9200 --name kibana --net=container:elasticsearch kibana:7.4.2

flag provided but not defined: --network