systemctl start docker#启动服务

docker run ubuntu:15.10 /bin/echo "Hello world"//会创建一个容器container,执行完后会关掉窗口

* **docker:** Docker 的二进制执行文件。
* **run:** 与前面的 docker 组合来运行一个容器。
* **ubuntu:15.10** 指定要运行的镜像，Docker 首先从本地主机上查找镜像是否存在，如果不存在，Docker 就会从镜像仓库 Docker Hub 下载公共镜像。
* **/bin/echo "Hello world":**在启动的容器里执行的命令

runoob@runoob:~$ docker run -i -t ubuntu:15.10 /bin/bash #退出是exit //会创建一个容器container,并且进入容器

* **-t:** 在新容器内指定一个伪终端或终端。
* **-i:** 允许你对容器内的标准输入 (STDIN) 进行交互。

Docker start 243c32535da7 //启动窗口

# docker exec -it 243c32535da7 /bin/bash//进行容器 //exit 后台一直在运行 docker stop 243c32535da7

docker attach 1e560fca3906（**注意：** 如果从这个容器退出(exit)，会导致容器的停止）

后台模式

docker run -d ubuntu:15.10 /bin/sh -c "while true; do echo hello world; sleep 1; done"

docker ps#查询后台列表(默认活动的 docker ps –a 所有)

//在宿主主机内使用 **docker logs** 命令，查看容器内的标准输出：

docker logs 2b1b7a428627

//停止

docker stop 4ac789d7df75

* **-d:** 后台运行

拉取镜像

docker pull ubuntu

启动某容器

docker start 2026a83f5f5e

导出镜像

docker export -o test-`date +%Y%m%d`.tar a404c6c174a2

//docker export 2026a83f5f5e >ubuntu.tar

**导入容器快照**

可以使用 docker import 从容器快照文件中再导入为镜像，以下实例将快照文件 ubuntu.tar 导入到镜像 test/ubuntu:v1:

docker import test-20210705.tar ubuntu:testv1

cat（英文全拼：concatenate）命令用于连接文件并打印到标准输出设备上

# 删除容器

docker rm -f 1e560fca3906

下面的命令可以清理掉所有处于终止状态的容器。

$ docker container prune

# 运行一个 web 应用

docker pull training/webapp # 载入镜像

docker run -d -p 5000:5000 training/webapp python app.py

docker port bf08b7f2cd89

docker logs -f bf08b7f2cd89

--检查进程

docker top 22c6768f2735

使用 **docker inspect** 来查看 Docker 的底层信息。它会返回一个 JSON 文件记录着 Docker 容器的配置和状态信息

# 列出镜像列表

我们可以使用 **docker images** 来列出本地主机上的镜像。

# 更新镜像

更新镜像之前，我们需要使用镜像来创建一个容器。

runoob@runoob:~$ docker run -t -i ubuntu:15.10 /bin/bash

root@e218edb10161:/#

在运行的容器内使用 **apt-get update** 命令进行更新。

在完成操作之后，输入 exit 命令来退出这个容器。

此时 ID 为 e218edb10161 的容器，是按我们的需求更改的容器。我们可以通过命令 docker commit 来提交容器副本。

runoob@runoob:~$ docker commit -m="has update" -a="runoob" e218edb10161 runoob/ubuntu:v2

sha256:70bf1840fd7c0d2d8ef0a42a817eb29f854c1af8f7c59fc03ac7bdee9545aff8

各个参数说明：

* **-m:** 提交的描述信息
* **-a:** 指定镜像作者
* **e218edb10161：**容器 ID
* **runoob/ubuntu:v2:** 指定要创建的目标镜像名
* 删除镜像
* docker rmi hello-world

docker run --privileged -it -d centos:centos7 /usr/sbin/init #生成一个新的容器

docker start mysqlbase

docker exec -it mysqlbase bash

mysql -h localhost -u root -p

mysql -h 127.0.0.1 -u root -p

mysql -h 127.0.0.1 -P 33061 -u root -p

我是boy2

CREATE USER 'lllogin'@'%' IDENTIFIED BY '123456';

GRANT ALL ON yaya.\* TO 'lllogin'@'%';

查找mysql配置

find -name my.cnf

无密码进入

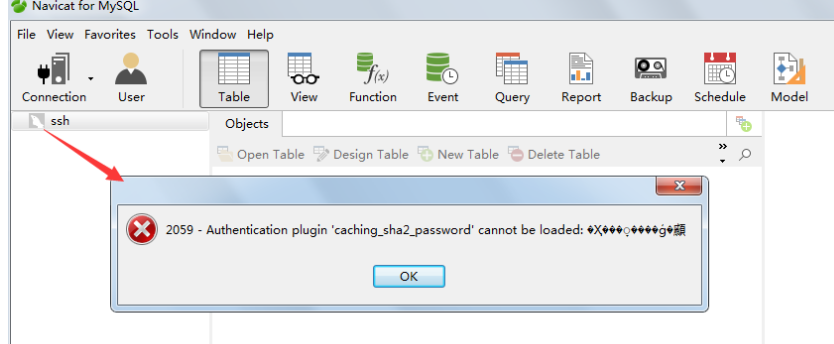
找到[mysqld],在它的后面任意下一行添加如下代码

skip-grant-tables

docker network connect eureka\_net mysqlbase --alias mysqlIP

utf8 -- UTF-8 Unicode

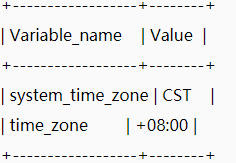
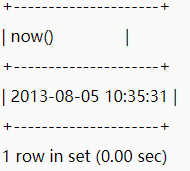
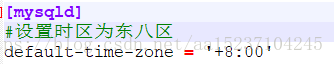
utf8\_general\_ci



alter user 'root'@'%' identified with mysql\_native\_password by 'Admin@123';

set password for 'root'@'localhost' = password('boy2');

根据网友Marksmanbat评论，如果在执行该步骤的时候出现ERROR 1290 (HY000): The MySQL server is running with the --skip-grant-tables option so it cannot execute this statement 错误。则执行下 flush privileges 命令，再执行该命令即可。

1. 设置为北京时间：
   1. mysql> set time\_zone='+8:00';
   2. mysql> Query OK, 0 rows affected (0.00 sec)
2. 再次检查mysql系统时区：
   1. mysql>show variables like '%time\_zone%';
   2. 
3. 再通过select now()来验证时区：
   1. mysql> select now();
   2. 
4. 找到my.ini, 在mysqld 下增加  default-time-zone = '+8:00'
5. 必须放到mysqld 下,放入其它位置无效
6. 

show global variables like 'port';

# 更改现有容器的接口 一定要停止docker,修改后再启动

docker inspect elasticsearch //查看启动设置

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

jq . ./hostconfig.json// jq [空格].[空格]/路径

# jq 查看json 格式文件

# 更改 HostPort 就是映射的宿主机端口。

只改了hostconfig.json另一个没改,成功了



2．/var/lib/docker/containers/[hash\_of\_the\_container]/ config.v2.json



**重点：此目录需要设置权限，否则启动容器报错权限错误**

#修改权限（1000:1000 是UID和GID）

chown -R 1000:1000 /home/jenkins/

# 查看IP

## 1.安装net-tools 命令：yum install net-tools -y

## 2.查看 docker0 inet addr后面的即为docker容器ip 命令：ifconfig

# Docker-Centos 安装JDK

yum search java|grep jdk

yum install java-1.8.0-openjdk

# [Docker] docker解决mysql连接慢的问题

## 配置文件路径:

/etc/mysql/mysql.conf.d

skip-name-resolve

//innodb\_flush\_log\_at\_trx\_commit=0

//sync\_binlog=0

## 运行容器，并且可以通过 exec 命令进入 CentOS 容器。

$ docker run -itd --name centos-test centos:centos7

docker exec -it centosbase /bin/bash

步骤1：先确认容器文件的默认路径

默认情况下为/var/lib/docker/containers

步骤2：进入容器目录

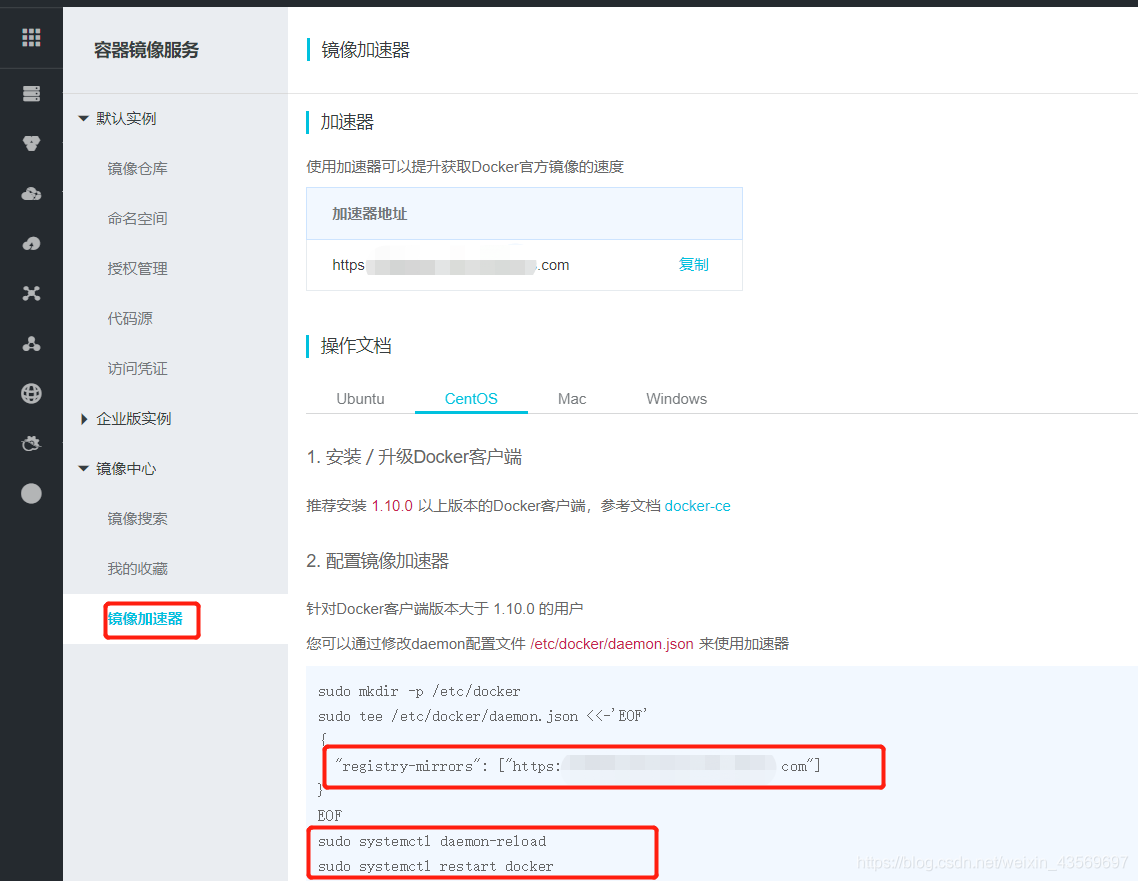
cd  /var/lib/docker/containers

步骤3：查看所有容器的大小  
du -sh \*

步骤3：查看某个容器的大小  
du -sh <容器id>

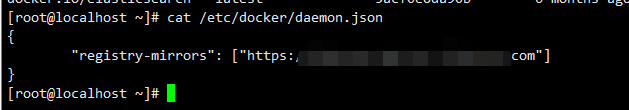
docker cp 宿主机文件路径  镜像名称:镜像中文件存放路径

## docker下载镜像卡死或太慢

找了网上很多方法，使用镜像中国也是下载卡死。  
最后一种成功了，下载速度很满意  
1.先再阿里云注册账号，不需要绑定什么，直接注册就可以了，有账号的直接登录就好。  
地址：https://cr.console.aliyun.com [点击跳转](https://cr.console.aliyun.com/)  
搜索 **容器镜像服务**  
  
2.创建镜像仓库  
  
3.创建好之后点到这个选项卡  
  
在终端输入

vim /etc/docker/daemon.json

* 1

将"registry-mirrors": ["https://......com"] （对应自己的加速地址）复制到文件中  
这是我的格式  
  
最后是重新加载文件和重启docker（不要忘记这两步）

sudo systemctl daemon-reload//重新加载

sudo systemctl restart docker//重启

* 1
* 2

最后就可以幸福的下载镜像了  
docker install mysql(还是这种命令，不需要有任何的修改)

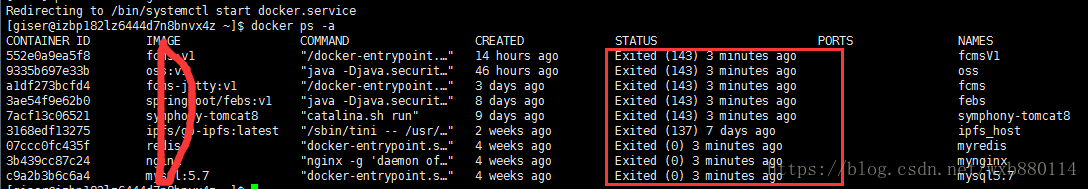
1、docker服务自动重启设置

systemctl enable docker.service

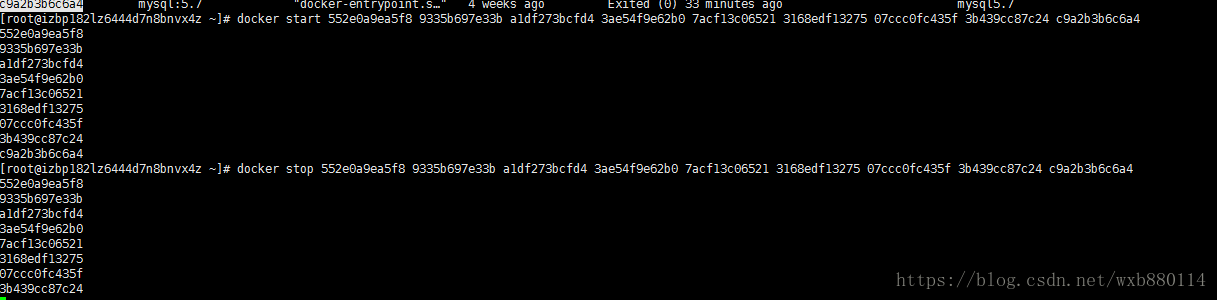
https://img-blog.csdn.net/20180930084945717?watermark/2/text/aHR0cHM6Ly9ibG9nLmNzZG4ubmV0L3d4Yjg4MDExNA==/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70

2、docker容器自动启动设置

docker ps -a



使用一下命令对镜像从起： docker restart imageid



在运行docker容器时可以加如下参数来保证每次docker服务重启后容器也自动重启：

## docker run --restart=always

如果已经启动了则可以使用如下命令：

## docker update --restart=always <CONTAINER ID>

日志

$ docker inspect --format='{{.LogPath}}' <容器ID>

/var/lib/docker/containers/545e06a75cc0ac8f8c1e6f7217455660187124a3eed031b5eb2f6f0edeb426cb/545e06a75cc0ac8f8c1e6f7217455660187124a3eed031b5eb2f6f0edeb426cb-json.log

echo > 日志路径

$ echo "" > $(docker inspect --format='{{.LogPath}}' <容器ID>)

echo "" > $(docker inspect --format='{{.LogPath}}' 00ffe61bf50f) #master

echo "" > $(docker inspect --format='{{.LogPath}}' 0ad8e8d298b9) #master

docker logs --tail=100 <容器ID>

查看日志

docker logs -f kibana

查找镜像

Docker search java

apt 包管理

镜像没有vi编辑器

1. apt update//只检查，不更新（已安装的软件包是否有可用的更新，给出汇总报告,如果失败可能是images版本太低了.不支持
2. apt-get install -y vim

# 安装

yum install -y git

docker run -d -i -t centos /bin/bash

Apache Calcite requires Java 1.8.0u202 or later. The current Java version is 1.8.0\_181

yum install wget

wget https://download.oracle.com/otn/java/jdk/8u202-b08/1961070e4c9b4e26a04e7f5a083f551e/jdk-8u202-linux-x64.tar.gz?AuthParam=1625499894\_9a1482f5bcaa1f953adb4cf50e091aa1

ls –lht

mkdir /usr/local/java/

tar -zxvf jdk-8u202-linux-x64.tar.gz\?AuthParam\=1625499894\_9a1482f5bcaa1f953adb4cf50e091aa1 -C /usr/local/java/

vim /etc/profile

export JAVA\_HOME=/usr/local/java/jdk1.8.0\_171

export JRE\_HOME=${JAVA\_HOME}/jre

export CLASSPATH=.:${JAVA\_HOME}/lib:${JRE\_HOME}/lib

export PATH=${JAVA\_HOME}/bin:$PATH

source /etc/profile

还没用过

ln -s /usr/local/java/jdk1.8.0\_171/bin/java /usr/bin/java

yum install vim

查看所有的版本

docker search logstash

docker search logstash --filter "is-official=true|false"

# docker 网络

<https://blog.csdn.net/wangyue23com/article/details/111172076>

## 1、docker network --help

#帮助命令后显示下面信息

[root@localhost /]# docker network --help

Usage: docker network COMMAND

Manage networks

Options:

--help Print usage

Commands:

connect Connect a container to a network

create Create a network

disconnect Disconnect a container from a network

inspect Display detailed information on one or more networks

ls List networks

prune Remove all unused networks

rm Remove one or more networks

Run 'docker network COMMAND --help' for more information on a command.

connect 将某个容器连接到一个docker网络

create 创建一个docker局域网络

disconnect 将某个容器退出某个局域网络

inspect 显示某个局域网络信息

ls 显示所有docker局域网络

prune 删除所有未引用的docker局域网络

rm 删除docker网络

## 2、创建自定义docker局域网络

docker network create mynet

然后pull两个镜像使用新创建的网络

## pull镜像和运行镜像

#pull之前先设置docker镜像加速，国外pull速度很慢，参考：

https://blog.csdn.net/wangyue23com/article/details/110194399

docker pull redis

docker pull nginx

#运行redis容器

docker run -itd --name redis --network mynet --network-alias redis -p 6379:6379 redis

#运行nginx容器

docker run -d --name nginx -p 80:80 --network mynet --network-alias nginx --privileged=true -v /home/wwwroot:/home/wwwroot -v /home/wwwlogs:/home/wwwlogs nginx

运行容器时候加入命令 --network mynet --network-alias redis，

代表当然容器要加入到mynet局域网络中

## 4、docker 查询网络信息

#查看mynet的详细信息，发现redis和nginx已在Containers容器内，属于当前mynet网络

[root@localhost /]# docker network inspect mynet

[

{

"Name": "mynet",

"Id": "8453b7b1d246f1ad54e27033928bbb8e7842ca96b4315fad720688af206b69c6",

"Created": "2020-12-13T22:50:55.566207522-08:00",

"Scope": "local",

"Driver": "bridge",

"EnableIPv6": false,

"IPAM": {

"Driver": "default",

"Options": {},

"Config": [

{

"Subnet": "172.18.0.0/16",

"Gateway": "172.18.0.1"

}

]

},

"Internal": false,

"Attachable": false,

"Containers": {

"87ff63bb819a823578bed89a67527009d9fee31a92de9d50e5d635b97cc18664": {

"Name": "redis",

"EndpointID": "19a04279300025dc936d206a287b1e31d04b339298b445694452a5f4326f7051",

"MacAddress": "02:42:ac:12:00:02",

"IPv4Address": "172.18.0.2/16",

"IPv6Address": ""

},

"c75ff4ac770fd04a1c6daa05a82533555038e3b0f8e15c3b1ee29e26c14a0131": {

"Name": "nginx",

"EndpointID": "4ccc014edcfd7be443c75491cc32aa0745561cfd45903e204e8d3e7afa94a4fe",

"MacAddress": "02:42:ac:12:00:03",

"IPv4Address": "172.18.0.3/16",

"IPv6Address": ""

}

},

"Options": {},

"Labels": {}

}

]

## 5、验证ping命令是否容器之间网络交互

#进入redis命令行，使用ping nginx命令验证能够连接成功\*\*使用ping命令之前，需要给docker安装ping，自行百度\*\*

docker exec -it redis bash

root@822f249e7d90:/# ping nginx

PING nginx (172.18.0.3): 56 data bytes

64 bytes from 172.18.0.3: icmp\_seq=0 ttl=64 time=0.071 ms

64 bytes from 172.18.0.3: icmp\_seq=1 ttl=64 time=0.100 ms

同样也可以不指定网络，直接启动容器，然后使用docker network connect接入网络。

#将容器删掉重新创建

docker stop nginx

docker rm -f nginx

docker stop redis

docker rm -f redis

## 使用docker network connect

#运行nginx和docker容器

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

docker run -d --name nginx -p 80:80 --privileged=true -v /home/wwwroot:/home/wwwroot -v /home/wwwlogs:/home/wwwlogs nginx

使用docker network connect命令连接网络

#加入nginx到mynet网络

[root@localhost /]# docker network connect mynet nginx

#加入redis到mynet网络

[root@localhost /]# docker network connect mynet redis

#查看mynet网络包含的容器，会发现两个容器都加入当前网络了，后面使用ping命令是能够ping通容器名称的。

[root@localhost /]# docker network inspect mynet

## docker network disconnect

#将nginx移除mynet局域网络

docker network disconnect mynet nginx

使用docker network 很方便的维护和管理docker网络，方便容器之间ip和端口交互。

# # 使用导入导出容器进行复制

docker export name/id > xxx.tar

docker import xxx.tar 新容器:v1

docker run -it 新容器:v1