## 一、安装Docker

### 第一步、卸载旧版本docker

sudo apt-get remove docker docker-engine docker.io containerd runc

### 第二步、更新及安装工具软件

1. 更新系统里的所有的能更新的软件

sudo apt-get update

1. 安装几个工具软件

sudo apt-get install apt-transport-https ca-certificates curl gnupg lsb-release

1. 增加一个docker的官方GPG key：

gpgkey：是用来验证软件的真伪 ——防伪的

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

1. 下载仓库文件

sudo add-apt-repository "deb [arch=amd64] https://mirrors.ustc.edu.cn/docker-ce/linux/ubuntu/ $(lsb\_release -cs) stable"

### 第三步、安装docker

1、再次更新系统

sudo apt-get update

2、安装docker-ce软件

sudo apt-get install docker-ce docker-ce-cli containerd.io -y

### 第四步、查看是否启动docker

由于docker安装的时候自带设置启动，所以直接查看进程是否启动就可以了

ps aux|grep docker

### 第五步、测试运行一个docker容器

sudo docker run hello-world

## 打包本地环境及代码

1. pip freeze > requirements.txt # 导出当前环境
2. pip download -d ./packs -r requirements.txt # 下载离线安装包放到名为packs的路径

1、安装一个miniconda基础的镜像

sudo docker pull continuumio/miniconda3

## nvidia-docker

distribution=$(. /etc/os-release;echo $ID$VERSION\_ID) \

&& curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | sudo apt-key add - \

&& curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.list | sudo tee /etc/apt/sources.list.d/nvidia-docker.list

sudo systemctl restart docker

sudo docker run --rm --gpus all nvidia/cuda:10.2-base-ubuntu16.04 nvidia-smi

安装了nvidia-docker,但进入容器后无法使用GPU：

# nvidia-docker：nvidia-container-toolkit的安装方式

docker run --gpus=all --rm nvidia/cuda:10.0-base nvidia-smi

# nvidia-docker2

docker run --runtime=nvidia -e NVIDIA\_VISIBLE\_DEVICES=all --rm nvidia/cuda:10.0-base nvidia-smi

or

nvidia-docker run -e NVIDIA\_VISIBLE\_DEVICES=all --rm nvidia/cuda:10.0-base nvidia-smi

## 普通用户使用

2. 假如docker组

docker安装之后默认会创建docker组，docker组拥有读取套接字权限，因此将该普通用户加入docker组，该组下所有用户有访问unix套接字权限，最后刷新用户的初始组即可。

添加docker用户组：

sudo groupadd docker

#将登陆用户加入到docker用户组中：

sudo gpasswd -a $USER docker

#更新用户组，当前终端生效，新开终端需要重新输入指令：

newgrp docker

# a表示所有用户都被赋予后面的权限，一次配置，之后任意开启终端都有效：

sudo chmod a+rw /var/run/docker.sock

搜索镜像：docker search

获取镜像： docker pull

查看镜像：docker images

删除镜像：docker rmi

重启docker：systemctl start docker

docker版本：docker -v

随机启动nginx 容器 随机端口 docker run -d -p 80:80 --name mynginx nginx -d表示后台运行 --name 别名

查看容器启动情况： docker ps

进入容器：docker exec -it id /bin/bash

查看日志情况：docker logs Name/ID

查看最近20行日志：docker logs -f --tail -20 id

显示一个运行的容器里面的进程信息：docker top id

停止容器：docker stop id

启动容器：docker start id

查看容器启动情况：docker ps

查询所有容器：docker ps -a

强制停止容器：docker kill id

删除容器：docker rm id

强制删除容器：docker rm -f id

查看容器日志：docker logs id

首先docker images 查看容器名称和别名

批量停止容器：docker rm -f $(docker ps -aq)

docker run -d -p 81:80 --name myngin1 nginx

查看docker的硬盘空间使用情况：docker system df

Linux下设置PYTHONPATH环境变量有三种方法：一种作用于当前终端，一种作用于当前用户，一种作用于所有用户。

1.作用于当前终端，直接当前终端输入命令

$ export PYTHONPATH=$PYTHONPATH:<你的要加入的路径>

$ export PYTHONPATH=$PYTHONPATH:/home/hadoop/MyBI

注1：'/home/hadoop/MyBI'是项目MyBI的项目名

注2：作用范围当前终端，一旦当前终端关闭或在另一个终端中，则无效。

注3：这种方式立即生效。

2.作用于当前用户，修改当前用户目录下的'~/.bashrc'文件

$ vi ~/.bashrc

加入内容：

export PYTHONPATH=$PYTHONPATH:/home/hadoop/MyBI

也可以加入多个路径，用分号分隔

export PYTHONPATH=$PYTHONPATH:<你的要加入的路径1>:<你的要加入的路径2>:等等

注1：需要执行如下命令后生效（或者注销后重新登陆）

$ source ~/.bashrc

3.作用于所有用户（需要root权限修改），修改'/etc/profile'文件

$ vi /etc/profile

加入内容：

export PYTHONPATH=$PYTHONPATH:/home/hadoop/MyBI

注1：需要执行如下命令后生效（或者注销后重新登陆）

$ source /etc/profile

\*4.如果修改PATH环境变量，也是像上面的三种方式操作