数据中台环境搭建（Pub版）

> 2021年6月6日, ©Brice(datawm.cn)

- Docker
- Docker-compose
- Anaconda
- Python
- jupyter
- Mysql
- Mongodb
- Redis
- Elastic Search
- Spark
- Hbase
- Nginx

数据中台环境搭建(目录)

Linux系统

组件安装

H2

H3

```
#安装组件命令：
#ubuntu下
apt-get install *** #ubuntu下，可以通过apt install yum来使用yum插入相关的包，有时需要apt-get update -y,apt-get upgrade -y
dpkg -L ** #ubuntu下查看软件的安装位置。
#centos下
yum install *** #一般加-y参数，dnf已有代替原有yum的趋势
yum update -y  #更新yum
yum grouplist  #显示系统包

#从网络下载文件
wget URL #从网络上自动下载文件,-c断电续传，-t设置重连次数0为无限
curl URL #请求 Web 服务器，-u 'bob:123'用户名密码，-d '1.txt'或-d'login=emma&pass=123' 以POST方式发送数据到URL，-X POST 指定请求的方法。

#查看系统版本：
cat /proc/version #ubuntu
cat /etc/redhat-release  #centos

#更改deb源（ubuntu)
```

```
mv  /etc/apt/sources.list /etc/apt/sources.list.bak #备份文件：将原本的镜像文件保存一份，
vim  /etc/apt/sources.list #编辑镜像列表，将如下内容复制到文件中。
#阿里云镜像
deb http://mirrors.aliyun.com/ubuntu/ xenial main
deb http://mirrors.aliyun.com/ubuntu/ xenial-updates main
deb http://mirrors.aliyun.com/ubuntu/ xenial universe
deb http://mirrors.aliyun.com/ubuntu/ xenial-updates universe
deb http://mirrors.aliyun.com/ubuntu/ xenial-security main
deb http://mirrors.aliyun.com/ubuntu/ xenial-security universe
#清华源
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-updates main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-backports main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-security main restricted universe multiverse
```

用户设置

H3

```
#用户设置
vim /etc/ssh/sshd_config  #修改ssd_condig，如设置：PasswordAuthentication yes
service sshd restart
vim /root/.ssh/authorized_keys  #修改公钥，将公钥内容复制到该文件；也可直接使用cat *.pub >> .ssh/authorized_keys导入公钥
vim /etc/default/useradd   #修改新增用户的默认文件夹
```

```
useradd [user_name] -g [group_name] -d [user_DIR] -s /bin/bash #新增用户，-d设置用户文件夹(使用-m自动建立登入目录），-s设置shell(或者-s /sbin/nologin)，-g设置组;
#userdel -r *  #删除用户,包括用户文件夹
passwd [user_name]  #设置用户密码
#usermod -d [user_DIR] [user_name] #修改用户的默认文件夹

#设置用户的root权限
vim /etc/sudoers
#在“root  ALL=(ALL)  ALL”这一行下面，再加入一行：
[user_name]  ALL=(ALL)     ALL
```

环境变量与别名设置等

H3

```
#设置环境变量
#Linux的环境配置一般在用户主目录下的.profile或.bashrc文件（推荐），系统的etc目录下profile文件或environment文件
#使用vim打开相应的配置文件，在最后一行增加PATH即可
PATH=yourpath1:yourpath2:...:$PATH
echo $PATH，或者env命令查看设置是否成功。

#设置命令别名(自定义命令)
vim ~/.bashrc  #打开配置文件
#在文件最后追加
alias cp='cp -i'
alias dls='docker ps  -a  --format "table {{.ID}} {{.Names}} {{.Status}} {{.Size}} {{.Image}} {{.Ports}} {{.Mounts}} "'
alias dps='docker ps  --format "table {{.ID}} {{.Names}} {{.Status}} {{.Ports}}"'
alias de=' dexec() { docker exec -it $1 /bin/bash; }; dexec'
alias ds=' dstart(){ docker start $1; }; dstart'
alias pspy='ps -ef | grep python'
alias lsp='port() { lsof -i:$1; }; port'

source ~/.bashrc  #重新加载配置，使别名生效
```

常用系统功能

H3

```
#设置定时启动（centos）
vim test.sh  #添加自动执行内容
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
***
chmod 755 test.sh  #修改文件属性为可执行
crontab -e   #可能需要yum install crontabs安装crontabs。编辑crontab，类似于vi操作vim /etc/crontab
20 23 * * * root /home/pe/test.sh
#上述内容表示每天晚上23:20使用root用户运行脚本test.sh,5个参数分别表示分钟，小时，天，月，周
service crond start  #启动服务
ntsysv #查看系统启动的服务。可能需要yum install -y ntsysv安装
chkconfig --level 35 crond on #设置开机启动

#vim使用(vi,gedit,nano也类似)
I #写入
esc #退出回到命令模式
:q #退出，
:q! #不保存退出
:wq #
dd ##在命令模式下输入dd删除行。
:/ #命令行模式下输入栏出现"/"，输入需要查找的关键字，回车；继续查找关键字，输入n，向前查找，输入N（大写）
:noh #取消高亮

#ubuntu中文文件名乱码问题
apt-get install convmv  #安装convmv:
cd / #转到需要处理的文件夹
convmv -f gbk -t utf8 -r --notest *

#安装Gnome桌面（centos）
dnf groupinstall "workstation"  ##安装Gnome，如果用"server with GUI"安装在8.2下会报错
systemctl enable gdm --now #启动Gnome桌面
```

```
systemctl status gdm

#安装并设置Xrdp（使用win的远程桌面连接mstsc打开，或者用MobaXter的RDP模式可以打开）
dnf install xrdp
systemctl enable xrdp --now
systemctl status xrdp
ss -tulpn| grep xrdp #查看活动的网络套接字
nano /etc/xrdp/xrdp.ini #配置xrdp，在最后增加一行“exec gnome-session”。其实可以不配置就可以运行
systemctl restart xrdp #重启服务
```

常见Linux命令

H3

```
Ubuntu18动态桌面开启 Ctrl+Alt+↓
Ubuntu切换工作区：选择您想切换到的窗口。 按 Ctrl+Alt+Shift+→ ，← ，↓ ，↑ 将窗口移动到相应工作区。

command & #后台执行命令
command |& tee -a output.txt #将终端的输出保存到文件中。
command Tab #目录名或文件名自动补全
script screen.log  #终端输入的命令被保存到screen.log文件中，然后执行exit停止保存：

Ctrl+Z #命令行时候退出回到命令行
Ctrl+c #退出/停止执行命令
Ctrl+L #清屏
clear #清除屏幕
ctrl+h #显示点号开头的隐藏文件（ubuntu的窗口界面下）

sudo su #以root登录，在命令前直接加sudo则以root方式执行该命令，su root，sudo -s -H
exit #退出root用户
shutdown -r now #立刻重启；shutdown -r 10 过10分钟自动重启，shutdown -r 20:35 在时间为20:35时候重启；shutdown -c取消重启
reboot #重启
halt #立刻关机（一般加-p 关闭电源）
poweroff#立刻关机
```

```
shutdown -h now #立刻关机；shutdown -h 10 10分钟后自动关机，shutdown -c取消关机

#查看信息
who #查看用户信息
cat /etc/passwd #查看用户列表。可以使用vi /etc/passwd修改用户的默认文件夹等
top #查看CPU和内存占用情况
netstat -anptl  #查看端口占用情况
ifconfig #查IP信息，类似windosws下的ipconfig
systemctl status #查看系统模块状态

ps -ef | grep 进程名称  #查看特定进程，-ef表示查看全格式的全部进程，-A显示所有进程
lsof -i:22 #查看占用端口的进程
ls -l /proc/进程号/exe  #查看进程文件所在位置
kill -9 <pid>  #解除端口占用
head -n 5（查看前几行）
tail -n 10 （查看末尾几行）

df -h #显示设备的状况，直接使用“!!”也是一样的效果
du -sh /data/*  #查看文件夹容量，与下一行的命令等效
du -h --max-depth=1 /home  #统计目录(或文件)所占磁盘空间的大小。显示/home文件夹的下一层文件夹
find / -size +100M |xargs ls -lh

cp -r [源路径/源文件] [目标路径/目标文件名] #参数-r为带文件夹复制，加参数-f为强制覆盖
mv oldname newname #移动文件夹或修改文件夹名称
rm -rf * #删除一个非空的文件夹
tar -zcvf *.tar.gz */* #注意前面是压缩包的文件名，后面是待压缩的文件夹或文件名。参数：c：create，t:list，x:extract，f:filename，v:volume，z:gzip
tar -xvf *.tar.gz #解压,也可以使用gzip -d，tar -x;
unzip #解压，需要先apt-get安装再使用。

ls -l #查看文件夹详细信息，缩写为ll。
ls -a |more #显示隐藏文件，more是分屏显示文件列表。

```

```
chmod -R 750 */*    #修改文件夹权限为读写执行，-R表示包括子文件夹，777为全部权限
chown 用户名:用户组 目录名/文件名 -R  #修改文件所属用户和租，R表示包括文件夹下内容
chgrp  用户组 文件名 -R
```

Docker

ubuntu安装

H2

```
apt-get install docker
apt-get install -y docker.io  #也是一个安装命令
```

H3

centos8安装

```
curl https://download.docker.com/linux/centos/docker-ce.repo -o /etc/yum.repos.d/docker-ce.repo  #下载docker-ce的repo
yum install https://download.docker.com/linux/fedora/30/x86_64/stable/Packages/containerd.io-1.2.6-3.3.fc30.x86_64.rpm  #安装依赖（这是相比centos7的关键步骤）
yum install docker-ce -y   #安装docker-ce
```

修改daemon配置文件

H3

```
#国内镜像站、https支持和更换images存储位置（daemon.json文件默认在/etc/docker/或/var/lib/docker）：
mkdir -p /etc/docker
tee /etc/docker/daemon.json <<-'EOF'
{
  "registry-mirrors": ["https://docker.mirrors.ustc.edu.cn"],
  "registry-mirrors": ["http://mirror.ccs.tencentyun.com"],
  "registry-mirrors": ["https://registry.docker-cn.com"],
  "insecure-registries":["0.0.0.0/0","[your-dns-or-ip]:8005"],
  "graph":"/home/docker"
}
EOF
systemctl daemon-reload
systemctl restart docker
```

docker常用命令

H3

```
systemctl enable docker  #设置自动启动docker
systemctl status docker
systemctl stop docker  #或者用 service docker stop
systemctl start docker  #或者用 service docker start
service docker restart
docker version
docker help

docker images  #查看本地已有镜像，加 -a 参数查看全部镜像（含中间层镜像）
docker search [--filter=stars=600]  [image-name]  #在仓库搜索镜像，--filter增加搜索条件，比如搜索starts大于600的镜像
docker pull [image-name]  #镜像下载
docker rmi [name/id]  #镜像删除
docker rmi -f [name/id]  [name2/id2]  #强制删除镜像，多个镜像以空格间隔
```

```
14 docker build -f  /docker/dockerfile/try01 -t try01:1.0  #构建docker镜像，先要编写相应的dockerfile，如：cd /docke/dockerfile  vim try01
15 docker commit -a="first commit" -m="my try01" [容器ID]  try01:v1.0 #基于当前容器生成一个新的镜像。参数：-a 作者；-c 使用dockerfile指令；-m :说明文字；-p :在commit时，将容器暂停
16
17 docker ps -a #查看运行的容器，-a 查看所有容器，-s 显示运行容器总文件大小
18 docker rename 原容器名称 新容器名称
19 docker run -itd --restart=always -p [port-h:port-c] --name [contain-name] --hostname [hostname] #以守护态启动容器，i交互模式，t分配一个伪输入终端，d 守护方式后台运行，--restart=always表示自动启动；  -p：将容器内部端口向外映射  --name：命名容器名称  -v：将容器内数据文件夹或者日志、配置等文件夹挂载到宿主机指定目录。
20 docker start [name/id]  #启动已停止的容器
21 docker restart [name/id]  #重启容器
22 docker top [name/id]  #查看容器中运行的进程。查看所有运行的容器的进程for i in  `docker ps |grep Up|awk '{print $1}'`;do echo \ &&docker top $i; done
23 docker exec -it [name/id] /bin/bash  #进入一个容器
24 docker attach [name/id] #直接进入容器终端，不启动新进程
25 exit  #关闭容器并退出。仅退出容器不关闭：快捷键：Ctrl + P + Q
26 docker stop [name/id] #停止一个运行中的容器
27 docker kill  [name/id] #杀掉一个运行中的容器
28 docker rm  [name/id] #删除一个已停止的容器，删除运行中的容器在rm后加-f，同时删除容器挂载的数据卷加-v
29
30 docker cp rabbitmq:/[container_path] [local_path]  #将rabbitmq容器中的文件copy至本地路径
31 docker cp [local_path] rabbitmq:/[container_path]/  #将主机文件copy至rabbitmq容器
```

```
docker cp [local_path] rabbitmq:/[container_path]  #将主机文件copy至rabbitmq容器，目录重命名为[container_path]（注意与非重命名copy的区别）

```

Docker-compose

H3

```
yum -y install python-pip
pip install --upgrade pip
pip install docker-compose  #可以通过pip安装
yum -y install docker-compose #centos下也可以直接安装
docker-compose --version

docker-compose -f ***.yml up -d   #Create and start containers。不加-f会自动运行docker-compose.yml文件中定义的服务。
docker-compose -f ***.yml stop #批量停止容器运行
docker-compose -f ***.yml start #批量开始容器运行
```

Python与Anaconda

安装anaconda

H2

H3

```
#下载anaconda安装包（清华资源）
wget https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/Anaconda3-2019.10-Linux-x86_64.sh
#bash安装
bash Anaconda3-2019.10-Linux-x86_64.sh -u
#输入命令后，一路enter，yes，在确认安装路径步骤时可以设置自定义文件夹，默认在~/anaconda3文件夹。
#如果想覆盖安装，需要加-u参数，并在确认安装路径时输入原来的路径

#docker安装anaconda和jupyter
docker pull continuumio/anaconda3
docker run -itd --name="anaconda" --hostname anaconda3 -p 8888:8888 -v [your-dir]:/root continuumio/anaconda3
docker exec -it anaconda /bin/bash
```

```
conda install -c conda-forge jupyterlab
jupyter notebook password  #设置密码
cd ~
jupyter lab --ip='*' --port=8888 --no-browser --allow-root &  # 用完以后Ctrl+C退出jupyter
exit
```

常用conda命令

> 环境名以“py36”为例，包名以“pandas”为例

H3

```
conda env list  #查看环境列表
conda activate py36 #激活环境
conda deactivate #退出当前环境；
conda config --set auto_activate_base false #或将auto_activate_base参数设置为false,那要进入的话通过conda activate base，如果反悔了可通过conda config --set auto_activate_base true来恢复。
conda create -n py36 python=3.6 #新建环境（可以指定Py版本）
conda create -n py36_new --clone py36 #复制环境（可用于环境改名）
conda remove -n py36 --all  #删除环境和其下的所有包
conda list  #查看当前环境所支持的包
conda remove pandas #卸载包
conda update pandas  #更新包
conda env export > environment.yaml  #使用 conda env create -f environment.yaml 创建环境（注意需要修改环境名和路径）
pip freeze >requirements.txt  #使用 pip install -r requirements.txt安装（需要先升级pip install --upgrade pip）

###添加conda镜像源：
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/
conda config --set show_channel_urls yes
```

安装python包

H3

```
#1)使用conda安装
conda install pandas
#2）通过pip安装
conda install pip
pip install --upgrade pip
pip install pandas==1.1.3
pip install -r requirements.txt
pip install -i https://pypi.tuna.tsinghua.edu.cn/simple  pandas #使用临时镜像安装
#3）本地有setup.py，到对应的文件夹下用setup安装
python setup.py install
#4）将包文件复制到文件夹下，在*.py文件中使用import引用
from *** import *** as ***
```

设置pip源

H3

常见的源：
清华：https://pypi.tuna.tsinghua.edu.cn/simple
阿里云：http://mirrors.aliyun.com/pypi/simple/
中国科技大学 https://pypi.mirrors.ustc.edu.cn/simple/
华中理工大学：http://pypi.hustunique.com/

linux下修改 ~/.pip/pip.conf (没有就创建一个文件夹及文件)
Windws下修改%APPDATA%/pip/目录下的配置文件（pip.ini），先按下win+R键，输入 %APPDATA%。一般是C:\Users.....\Appdata\roming 文件夹，若没有pip/pip.ini，则新建一个，
输入如下内容之一：

```
#清华
[global]
time-out=60
index-url =
https://pypi.tuna.tsinghua.edu.cn/simple
#阿里云
[global]
index-
url=http://mirrors.cloud.aliyuncs.com/pypi/simple
/
[install]
trusted-host=mirrors.cloud.aliyuncs.com
```

Jupyter

H3

```
#windows下需要先安装python解释器，conda下先切换到特定环境。
pip install jupyterlab #或者pip install jupyter notebook
jupyter notebook password    # 访问密码重置
#切换到特定文件夹、特定环境下执行下面三条命令中的一条
jupyter lab  #启动notebook服务
python -m notebook #有时可用python命令来启动notebook服务
nohup jupyter notebook --port 6666 --ip 0.0.0.0 --no-browser --allow-root &  #后台运行，远程访问6666
#然后根据提示在浏览器中使用jupyter notebook
#端口号可根据需要自己设定,未设置密码的话可用如下命令查看token
jupyter notebook list
#结束服务 ctrl + c
```

数据库

Mysql（通过docker安装）

H2

H3

```
docker  pull mysql
docker run -d -p 3306:3306  -v [your-mysql-dir]:/var/lib/mysql --hostname mysql  --name mysql -e MYSQL_ROOT_PASSWORD=*  mysql
docker exec -it mysql /bin/bash  #进入容器内部

mysql -uroot -p  #打开mysql命令行
```

> docker--在mysql命令行模式下：

```
mysql>alter user 'root'@'%' identified with mysql_native_password by 'root';   --允许root用户远程访问，这句话好像不设置也不会出错。
mysql>select version();   --查看mysql版本e

--增加远程用户：
mysql>create user 'mysql'@'%' identified by '[your-passwd]';
mysql>GRANT all ON *.* TO 'mysql'@'%';
mysql>flush privileges;

mysql>create database test1;
mysql>create user 'user1'@'%' identified by '[your-passwd]'
mysql>use test1;
mysql>GRANT select ON test1.* TO 'user1'@'%';
mysql>exit
```

Mongo（通过docker安装）

H3

```
docker pull mongo
docker run -itd --hostname mongo --name mongo -p 27017:27017 -v [your-mongo-dir]:/data/db mongo --auth  #--atuh是指需要密码才能访问
docker exec -it mongo mongo admin  #进入容器内部mongo命令行
```

> 在mongo命令行模式下：

```
--创建一个名为 root，密码为***的用户。
>  db.createUser({ user:'root',pwd:'[your-passwd]',roles:[ { role:'userAdminAnyDatabase', db: 'admin'}]});
--尝试使用上面创建的用户信息进行连接。
> db.auth('root', '[your-passwd]')

--新建库和用户：
> use [dbname]  --库名
> db.foo.insert({_id:1,name:"test"})  --临时插入一个集合，保持库存在。或者使用db.usr.insert({'name':'tompig'});
> show collections
> db.createUser({ user:'[username]', pwd:'[passwd]', roles:['readWrite'] })  --创建用户前需要先登录root用户。readWrite可以是dbOwner
> db.auth('[username]','[passwd]')
> exit
```

Redis（通过docker安装）

H3

```
#新建/data/Lake/redis/conf/文件夹，并切换到该文件夹
wget http://download.redis.io/redis-stable/redis.conf #下载conf文件并修改
docker pull redis
docker run -d -p 6379:6379 --hostname redis  --name redis -v [your-redis-dir]/data:/data  -v [your-redis-dir]/conf/redis.conf:/etc/redis/redis.conf redis redis-server /etc/redis/redis.conf --appendonly yes
docker exec -it redis /bin/bash
```

> 修改conf文件：

```
bind 127.0.0.1 #注释掉这部分，这是限制redis只能本地访问
protected-mode no #默认yes，开启保护模式，限制为本地访问
daemonize no #默认no，改为yes意为以守护进程方式启动，可后台运行，除非kill进程，改为yes会使配置文件方式启动redis失败
databases 16 #数据库个数（可选），我修改了这个只是查看是否生效
stop-writes-on-bgsave-error yes  建议改为no，RASA写对话时不改会出错。
dir  ./ #输入本地redis数据库存放文件夹（可选）
appendonly yes #redis持久化（可选）
```

neo4j安装

H3

```
docker search neo4j
docker pull neo4j
#建立四个基本的文件夹
mkdir [your-neo4j-dir]/data #数据存放的文件夹
mkdir [your-neo4j-dir]/logs #运行的日志文件夹
mkdir [your-neo4j-dir]/conf #数据库配置文件夹（在配置文件neo4j.conf中配置包括开放远程连接、设置默认激活的数据库）
mkdir [your-neo4j-dir]/import #为了大批量导入csv来构建数据库，需要导入的节点文件nodes.csv和关系文件rel.csv需要放到这个文件夹下）

docker run -d --name container_name -p 7474:7474 -p 7687:7687 -v [your-neo4j-dir]/data:/data -v [your-neo4j-dir]/logs:/logs -v [your-neo4j-dir]/conf:/var/lib/neo4j/conf -v [your-neo4j-dir]/import:/var/lib/neo4j/import --env NEO4J_AUTH=neo4j/password neo4j

#在浏览器中输入localhost:7474就可以登陆数据库了

#设置远程登陆等
cd [your-neo4j-dir]/conf
vim neo4j.conf
#在文件配置末尾添加这一行
```

```
dbms.connectors.default_listen_address=0.0.0.0 #指定连接器的默认监听ip为0.0.0.0，即允许任何ip连接到数据库
#在文件配置修改
dbms.connector.bolt.listen_address=0.0.0.0:7687 #取消注释并把对bolt请求的监听“地址:端口”改为“0.0.0.0:7687”
dbms.connector.http.listen_address=0.0.0.0:7474 #取消注释并把对http请求的监听“地址:端口”改为“0.0.0.0:7474”
#保存后退出，重启neo4j容器，可以使用容器的省略id或者生成容器时指定的容器名进行重启。
docker restart 容器id（或者容器名）
```

大数据

HBase

H2

H3

```
docker search hbase
docker pull harisekhon/hbase
docker images
docker run -d  -p 2181:2181 -p 8080:8080 -p 8085:8085 -p 9090:9090 -p 9095:9095 -p 16000:16000 -p 16010:16010 -p 16201:16201 -p 16301:16301 -v [your-hbase-dir]:/data --name hbase harisekhon/hbase:latest  #如果容器还未创建
docker start hbase #如果容器已经存在
docker exec -it hbase /bin/bash
hbase shell
```

> 在hbase shell提示符下：

```
list  --显示当前库中的表
create 'user1','personalinfo'  --创建hbase表
put 'user1','roow1','personalinfo:name','zhangsan' --插入
get 'user1','row1','personalinfo:name'  --查询
help  --帮助查看其他命令
```

---

sparks

> 使用docker-compose快速安装sparks集群（centos）

H3

```
docker-compose --version

docker-compose -f docker-spark.yml up -d
docker exec -it spark /bin/bash
spark-shell  #退出用exit命令
```

> 在hbase shell提示符下：

```
:quit  --退出shell
```

> 【配置文件】docker-spark.yml

```
cat << EOF >docker-spark.yml
version: '2'
services:
  namenode:
    image: uhopper/hadoop-namenode:2.8.1
    # 配置好 docker 内的假域名
    hostname: namenode
    container_name: namenode
    networks:
      - hadoop
    volumes:
      # 自行修改数据卷的映射位置
      - [your-spark-dir]/namenode:/hadoop/dfs/name
    environment:
      - CLUSTER_NAME=datanode1
      - CLUSTER_NAME=datanode2
      - CLUSTER_NAME=datanode3
      # 配置 hdfs 用户权限问题,不需要只允许 hadoop 用户访问
      - HDFS_CONF_dfs_permissions=false
    ports:
      # 接收Client连接的RPC端口，用于获取文件系统metadata信息
      - 8020:8020
      # nameNode http服务的端口
      - 50070:50070
```

```
      # nameNode https 服务的端口
      - 50470:50470
  datanode1:
    image: uhopper/hadoop-datanode:2.8.1
    hostname: datanode1
    container_name: datanode1
    networks:
      - hadoop
    volumes:
      - [your-spark-dir]/datanode1:/hadoop/dfs/data
    environment:
      # 等价于在 core-site.xml 中配置 fs.defaultFS
      - CORE_CONF_fs_defaultFS=hdfs:#namenode:8020
      # 等价于在 hdfs-site.xml 中配置 dfs.datanode.address
      - HDFS_CONF_dfs_datanode_address=0.0.0.0:50010
      # dfs.datanode.ipc.address 不使用默认端口的意义是在同一机器起多个 datanode，暴露端口需要不同
      - HDFS_CONF_dfs_datanode_ipc_address=0.0.0.0:50020
      # dfs.datanode.http.address
      - HDFS_CONF_dfs_datanode_http_address=0.0.0.0:50075
    ports:
      - 50010:50010
      - 50020:50020
      - 50075:50075
  datanode2:
    image: uhopper/hadoop-datanode:2.8.1
    hostname: datanode2
    container_name: datanode2
    networks:
      - hadoop
    volumes:
```

```
      - [your-spark-dir]/datanode2:/hadoop/dfs/data
    environment:
      - CORE_CONF_fs_defaultFS=hdfs:#namenode:8020
      - HDFS_CONF_dfs_datanode_address=0.0.0.0:50012
      - HDFS_CONF_dfs_datanode_ipc_address=0.0.0.0:50022
      - HDFS_CONF_dfs_datanode_http_address=0.0.0.0:50072
    ports:
      - 50012:50012
      - 50022:50022
      - 50072:50072
  datanode3:
    image: uhopper/hadoop-datanode:2.8.1
    hostname: datanode3
    container_name: datanode3
    networks:
      - hadoop
    volumes:
      - [your-spark-dir]/datanode3:/hadoop/dfs/data
    environment:
      - CORE_CONF_fs_defaultFS=hdfs:#namenode:8020
      - HDFS_CONF_dfs_datanode_address=0.0.0.0:50013
      - HDFS_CONF_dfs_datanode_ipc_address=0.0.0.0:50023
      - HDFS_CONF_dfs_datanode_http_address=0.0.0.0:50073
    ports:
      - 50013:50013
      - 50023:50023
      - 50073:50073
```

```
  resourcemanager:
    image: uhopper/hadoop-resourcemanager:2.8.1
    hostname: resourcemanager
    container_name: resourcemanager
    networks:
      - hadoop
    environment:
      -
CORE_CONF_fs_defaultFS=hdfs:#namenode:8020
      -
YARN_CONF_yarn_log___aggregation___enable=true
    ports:
      - 8030:8030
      - 8031:8031
      - 8032:8032
      - 8033:8033
      - 8088:8088
  nodemanager:
    image: uhopper/hadoop-nodemanager:2.8.1
    hostname: nodemanager
    container_name: nodemanager
    networks:
      - hadoop
    environment:
      -
CORE_CONF_fs_defaultFS=hdfs:#namenode:8020
      -
YARN_CONF_yarn_resourcemanager_hostname=resourc
emanager
      -
YARN_CONF_yarn_log___aggregation___enable=true
      -
YARN_CONF_yarn_nodemanager_remote___app___log__
_dir=/app-logs
    ports:
      - 8040:8040
      - 8041:8041
      - 8042:8042
  spark:
    image: uhopper/hadoop-spark:2.1.2_2.8.1
    hostname: spark
```

```
115     container_name: spark
116     networks:
117       - hadoop
118     environment:
119       -
    CORE_CONF_fs_defaultFS=hdfs:#namenode:8020
120       -
    YARN_CONF_yarn_resourcemanager_hostname=resourc
    emanager
121     command: tail -f /var/log/dmesg
122     ports:
123       - 4040:4040
124 networks:
125   hadoop:
126
127 EOF
128 #[spark 集群配置文件结束]
```

ElasticSearch/Kibana

H3

```
1 #远程访问
2 http://[your-dns-or-ip]:9200
3 http://[your-dns-or-ip]:9200/_cat/nodes?v  #查看
  集群的节点信息
4 http://[your-dns-or-ip]:9200/_cat/indices?v  #查
  看索引信息
5
6 http://[your-dns-or-ip]:5601
7 http://[your-dns-or-ip]:5601/status  #查看kibana
  状态
```

```
docker-compose -f docker-es.yml up -d
docker  ps
#查看ES集群安装情况，应该有es01,es02,es03三个容器启动。

#执行命令
sysctl -w vm.max_map_count=262144
sysctl -a|grep vm.max_map_count
#永久保存
echo 'vm.max_map_count=262144' > /etc/sysctl.conf
sysctl -p

curl 'http://localhost:9200/'
curl -X GET "localhost:9200/_cat/nodes?v&pretty"
```

【配置文件】docker-es.yml

```
cat << EOF >docker-es.yml
version: '2.2'
services:
  es01:
    image: docker.elastic.co/elasticsearch/elasticsearch:7.9.1
    container_name: es01
    environment:
      - node.name=es01
      - cluster.name=es-docker-cluster
      - discovery.seed_hosts=es02,es03
      - cluster.initial_master_nodes=es01,es02,es03
      - bootstrap.memory_lock=true
      - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
    ulimits:
      memlock:
        soft: -1
        hard: -1
    volumes:
      - [your-es-dir]/data01:/usr/share/elasticsearch/data
    ports:
```

```
      - 9200:9200
    networks:
      - elastic
  es02:
    image:
docker.elastic.co/elasticsearch/elasticsearch:7.
9.1
    container_name: es02
    environment:
      - node.name=es02
      - cluster.name=es-docker-cluster
      - discovery.seed_hosts=es01,es03
      -
cluster.initial_master_nodes=es01,es02,es03
      - bootstrap.memory_lock=true
      - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
    ulimits:
      memlock:
        soft: -1
        hard: -1
    volumes:
      - [your-es-
dir]/data02:/usr/share/elasticsearch/data
    networks:
      - elastic
  es03:
    image:
docker.elastic.co/elasticsearch/elasticsearch:7.
9.1
    container_name: es03
    environment:
      - node.name=es03
      - cluster.name=es-docker-cluster
      - discovery.seed_hosts=es01,es02
      -
cluster.initial_master_nodes=es01,es02,es03
      - bootstrap.memory_lock=true
      - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
    ulimits:
      memlock:
        soft: -1
```

```
        hard: -1
    volumes:
      - [your-es-dir]/data03:/usr/share/elasticsearch/data
    networks:
      - elastic
  kibana:
    image: docker.elastic.co/kibana/kibana:7.9.1
    container_name: kibana
    volumes:
      - [your-es-dir]/kibana/kibana.yml:/usr/share/kibana/config/kibana.yml
    ports:
      - 5601:5601
    networks:
      - elastic
volumes:
  data01:
    driver: local
  data02:
    driver: local
  data03:
    driver: local

networks:
  elastic:
    driver: bridge
EOF
#可以配置“port:-对外:对内”的对外的端口，比如49200，
#此外，需要chmod -R 777 [your-es-dir]
#需要预先设定号kibana.yml配置文件（配置方法见下一节）
#kibana需要在谷歌浏览器或者火狐浏览器打开，IE浏览器有时候不行。
```

工具软件

VSCode

快捷键 Ctrl+Shift+P：打开交互搜索框

交互模式：在每块cell前加上 #%% ，就可以按cell执行代码，也有shift+enter的实行cell

H3

常见插件：

```
中文：Chinese （Simplified）Language Pack for Visual Studio Code
自动完成另一侧标签的同步修改：Auto Rename Tag
格式化代码：Beautify
给括号加上不同的颜色：Bracket Pair Colorizer
万能语言运行环境： Code Runner
```

Nginx与反向代理设置

H3

```
docker pull nginx
mkdir [web_DIR]  #将要部署的文件夹复制到这里
docker run -p 80:80 -v [web_DIR]:/usr/share/nginx/html --hostname nginx  --name nginx  -d   nginx
#docker run -p 80:80 -p 443:443 -v [web_DIR]:/usr/share/nginx/html -v [config_DIR]:/usr/share/nginx/config --hostname nginx  --name nginx -d nginx  #完整的命令
service nginx stop #停止
service nginx start #启动
service nginx restart #重启
```

nginx的配置文件有两个，一个是/etc/nginx/nginx.conf，一个是/etc/nginx/conf.d/default.conf
修改配置文件，一是将文件复制到映射的文件夹，修改后使用cp命令覆盖。二是在容器内使用apt-get install vim安装编辑器。

Https的配置，
需要先去阿里云或者腾讯云申请一个免费的ssl证书，或者使用acme.sh自助生成，，
下载证书保存在[config_DIR] 或[config_DIR]/cert文件夹下。

> nginx配置文件示例如下：

```
server {
    listen 443 ssl;
```

```
    server_name datawm.cn;
    #ssl on;
    ssl_certificate /etc/nginx/cert/8888888_datawm.cn.pem;
    ssl_certificate_key /etc/nginx/cert/8888888_datawm.cn.key;
    ssl_session_timeout 5m;
    ssl_ciphers ECDHE-RSA-AES128-GCM-SHA256:ECDHE:ECDH:AES:HIGH:!NULL:!aNULL:!MD5:!ADH:!RC4;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_prefer_server_ciphers on;

    location / {
        root   /usr/share/nginx/html;
        index  index.html index.htm;
    }
    #配置反向代理，可使用docker inspect [container_name]  #查看某个容器的IP地址
    location /newApi/{
        rewrite ^/newApi/(.*)$ /$1 break;
        proxy_pass http://172.17.0.2:8080;
      }
    #error_page  404              /404.html;
    # redirect server error pages to the static page /50x.html
    #
    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }
}

server {
    listen       80;
    listen  [::]:80;
    server_name  datawm.cn;
    rewrite ^(.*)$ https://$host$1 permanent;
}
```

END