# Minio安装

root/[Admin@2019.com](mailto:Admin@2019.com)

| 主机名 | IP | 存储 |
| --- | --- | --- |
| minio-server1 | 10.8.251.32 | /opt/minio/data/export{1,2} |
| minio-server2 | 10.8.251.33 | /opt/minio/data/export{1,2} |
| minio-server3 | 10.8.251.34 | /opt/minio/data/export{1,2} |

### 端口

| IP | 功能 | 端口 | 描述 |
| --- | --- | --- | --- |
| 10.8.251.32/10.8.251.33/10.8.251.34 | API端口 | 9000 |  |
| 10.8.251.32/10.8.251.33/10.8.251.34 | 控制台端口 | 9001 | 用户名密码：admin/admin@2022 |
| 10.8.251.32 | java访问负载API端口 | 19000 |  |
| 10.8.251.32 | 控制台负载端口 | 19001 | 用户名密码：admin/admin@2022 |

### 下载二进制包部署

mkdir -p /opt/minio ; cd /opt/minio  
wget https://dl.min.io/server/minio/release/linux-amd64/minio  
chmod +x /opt/minio

### 环境变量配置

# 加在/etc/profile  
export PATH=$PATH:/opt/minio

### 构造虚拟磁盘

#### 说明：系统盘只有50G,mino至少需要4块盘，三台机器每台机器虚拟2块硬盘。后期可以挂载硬盘替换虚拟磁盘。

（1）创建空的磁盘镜像文件，这里创建一个20G的虚拟盘

dd if=/dev/zero of=disk1.img bs=1M count=20000  
dd if=/dev/zero of=disk2.img bs=1M count=20000

（2）使用 losetup将磁盘镜像文件虚拟成块设备

losetup /dev/loop1 disk1.img  
losetup /dev/loop2 disk2.img

（3）挂载块设备

mount /dev/loop1 /opt/minio/data/export1  
mount /dev/loop2 /opt/minio/data/export2

### fstab添加

/dev/loop1 /opt/minio/data/export1 ext4 defaults 0 0   
/dev/loop2 /opt/minio/data/export2 ext4 defaults 0 0

### 所有节点运行脚本(minio.sh)

#!/bin/bash  
# 创建日志存储目录  
mkdir -p /opt/minio/logs  
# 分别在三个节点上创建存储目录  
mkdir -p /opt/minio/data/export{1,2}  
# 创建配置目录  
mkdir -p /etc/minio  
export MINIO\_ROOT\_USER=admin  
export MINIO\_ROOT\_PASSWORD=admin@2022  
  
# 在三台机器上都执行该文件，即以分布式的方式启动了MINIO  
# --address "0.0.0.0:9000" 挂载9001端口为api端口（如Java客户端）访问的端口  
# --console-address ":9000" 挂载9000端口为web端口；   
/opt/minio/minio server --address 0.0.0.0:9000 --console-address 0.0.0.0:9001 --config-dir /etc/minio \  
http://minio-server1/opt/minio/data/export1 \  
http://minio-server1/opt/minio/data/export2 \  
http://minio-server2/opt/minio/data/export1 \  
http://minio-server2/opt/minio/data/export2 \  
http://minio-server3/opt/minio/data/export1 \  
http://minio-server3/opt/minio/data/export2 > /opt/minio/logs/minio\_server.log

### 创建系统启动服务

cat > /usr/lib/systemd/system/minio.service <<EOF  
[Unit]  
Description=Minio service  
Documentation=https://docs.minio.io/  
  
[Service]  
WorkingDirectory=/opt/minio  
ExecStart=/opt/minio/minio.sh  
  
Restart=on-failure  
RestartSec=5  
  
[Install]  
WantedBy=multi-user.target  
EOF

##### 修改文件权限

chmod +x /usr/lib/systemd/system/minio.service   
chmod +x /opt/minio/minio   
chmod +x /opt/minio/minio.sh

### 启动集群

#重新加载服务  
systemctl daemon-reload  
#启动服务  
systemctl start minio  
#加入自启动  
systemctl enable minio

### web访问地址

http://10.8.251.32:9001/login  
http://10.8.251.33:9001/login  
http://10.8.251.34:9001/login

### nginx负载均衡

upstream minio\_api {  
 server 10.8.251.32:9000;  
 server 10.8.251.33:9000;  
 server 10.8.251.34:9000;  
}  
  
upstream minio\_console {  
 server 10.8.251.32:9001;  
 server 10.8.251.33:9001;  
 server 10.8.252.34:9001;  
}  
  
server{  
 listen 19000;  
 server\_name 10.8.252.32;  
 ignore\_invalid\_headers off;  
 client\_max\_body\_size 0;  
 proxy\_buffering off;  
 location / {  
 proxy\_set\_header X-Forwarded-Proto $scheme;  
 proxy\_set\_header Host $http\_host;  
 proxy\_set\_header X-Real-IP $remote\_addr;  
 proxy\_connect\_timeout 300;  
 proxy\_http\_version 1.1;  
 chunked\_transfer\_encoding off;  
 proxy\_ignore\_client\_abort on;  
 proxy\_pass http://minio\_api;  
 }  
}  
  
server{  
 listen 19001;  
 server\_name 10.8.251.32;  
 ignore\_invalid\_headers off;  
 client\_max\_body\_size 0;  
 proxy\_buffering off;  
 location / {  
 proxy\_set\_header X-Forwarded-Proto $scheme;  
 proxy\_set\_header Host $http\_host;  
 proxy\_set\_header X-Real-IP $remote\_addr;  
 proxy\_connect\_timeout 300;  
 proxy\_http\_version 1.1;  
 chunked\_transfer\_encoding off;  
 proxy\_ignore\_client\_abort on;  
 proxy\_pass http://minio\_console;  
 }  
}

### 负载访问地址

http://10.8.251.32:19001  
http://10.8.251.32:19000

python测试脚本

from minio import Minio  
from minio.error import S3Error  
try:  
 client = Minio('10.8.251.32:19000',access\_key='admin',secret\_key='admin@2022',secure=False)  
 found = client.bucket\_exists("yfl")  
 client.fput\_object("yfl", "2.py", "/opt/get-pip.py.1")  
except S3Error as e:  
 print("error:", e)  
print(found)# 返回布尔值 True or False

# dolphinscheduler集群环境部署

增加yfl用户

useradd yfl  
passwd yfl 123456

修改免密

vim /etc/sudoers  
yfl ALL=(ALL) NOPASSWD:ALL

注释掉不需要环境变量

/opt/dolphinscheduler/conf/env/dolphinscheduler\_env.sh

#export HADOOP\_HOME=/opt/soft/hadoop  
#export HADOOP\_CONF\_DIR=/opt/soft/hadoop/etc/hadoop  
#export SPARK\_HOME1=/opt/soft/spark1  
#export SPARK\_HOME2=/opt/soft/spark2  
#export PYTHON\_HOME=/opt/soft/python  
export JAVA\_HOME=/opt/jdk1.8.0\_341/  
#export HIVE\_HOME=/opt/soft/hive  
#export FLINK\_HOME=/opt/soft/flink  
export DATAX\_HOME=/opt/soft/datax  
  
#export   
export PATH=$JAVA\_HOME/bin:$DATAX\_HOME/bin:$PATH

### zookeeper 集群

| 主机名 | ip | myid | 功能服务 |
| --- | --- | --- | --- |
| node-server1 | 10.8.251.37 | 1 | zookeeper |
| node-server2 | 10.8.251.38 | 2 | zookeeper |
| node-server3 | 10.8.251.39 | 3 | zookeeper |

配置文件修改zoo.cfg

tickTime=2000 #通信心跳时间，Zookeeper服务器与客户端心跳时间，单位毫秒  
initLimit=10 #Leader和Follower初始连接时能容忍的最多心跳数（tickTime的数量），这里表示为10\*2s  
syncLimit=5 #Leader和Follower之间同步通信的超时时间，这里表示如果超过5\*2s，Leader认为Follwer死掉，并从服务器列表中删除Follwer  
dataDir=/opt/zookeeper-3.5.7/data #修改，指定保存Zookeeper中的数据的目录，目录需要单独创建  
dataLogDir=/opt/zookeeper-3.5.7/logs #添加，指定存放日志的目录，目录需要单独创建  
clientPort=2181 #客户端连接端口  
server.1=10.8.251.37:3188:3288  
server.2=10.8.251.38:3188:3288  
server.3=10.8.251.39:3188:3288

每个节点的dataDir指定的目录下创建一个 myid 的文件

echo 1 > /opt/zookeeper-3.5.7/data/myid  
echo 2 > /opt/zookeeper-3.5.7/data/myid  
echo 3 > /opt/zookeeper-3.5.7/data/myid

配置 Zookeeper 启动脚本

vim /etc/init.d/zookeeper

#!/bin/bash  
#chkconfig:2345 20 90  
#description:Zookeeper Service Control Script  
ZK\_HOME='/opt/zookeeper-3.5.7'  
case $1 in  
start)  
 echo "---------- zookeeper 启动 ------------"  
 $ZK\_HOME/bin/zkServer.sh start  
;;  
stop)  
 echo "---------- zookeeper 停止 ------------"  
 $ZK\_HOME/bin/zkServer.sh stop  
;;  
restart)  
 echo "---------- zookeeper 重启 ------------"  
 $ZK\_HOME/bin/zkServer.sh restart  
;;  
status)  
 echo "---------- zookeeper 状态 ------------"  
 $ZK\_HOME/bin/zkServer.sh status  
;;  
\*)  
 echo "Usage: $0 {start|stop|restart|status}"  
esac

设置开机启动

chmod +x /etc/init.d/zookeeper  
chkconfig --add zookeeper

启动服务

service zookeeper start

zookeeper测试

./zkCli.sh -server 127.0.0.1:2181  
create /zk  
get /zk  
set /zk "ssl"  
get /zk

## dolphinscheduler 安装

| 主机名 | ip | 功能服务 |
| --- | --- | --- |
| node-server1 | 10.8.251.37 | master work alert api |
| node-server2 | 10.8.251.38 | master work alert api |
| node-server3 | 10.8.251.39 | zookeeper work mysql |

### mysql安装

rpm --import https://repo.mysql.com/RPM-GPG-KEY-mysql-2022  
mysql install:  
rpm -ivh http://mirrors.ustc.edu.cn/mysql-repo/mysql57-community-release-el7.rpm  
yum install mysql mysql-server

mysql 用户名密码

root/123456

dolphinscheduler 安装配置

/opt/dolphinscheduler/conf/config

ips="node-server1,node-server2,node-server3"  
masters="node-server1,node-server2"  
workers="node-server1:default,node-server2:default,node-server3:default"  
alertServer="node-server3"  
apiServers="node-server1"  
pythonGatewayServers="node-server1"  
deployUser="root"  
javaHome="/opt/jdk1.8.0\_341/"  
apiServerPort="12345"  
DATABASE\_TYPE=${DATABASE\_TYPE:-"mysql"}  
SPRING\_DATASOURCE\_URL=${SPRING\_DATASOURCE\_URL:-"jdbc:mysql://node-server3:3306/dolphinscheduler?useUnicode=true&characterEncoding=UTF-8"}  
SPRING\_DATASOURCE\_USERNAME=${SPRING\_DATASOURCE\_USERNAME:-"root"}  
SPRING\_DATASOURCE\_PASSWORD=${SPRING\_DATASOURCE\_PASSWORD:-"123456"}  
registryServers="10.8.251.37:2181,10.8.251.38:2181,10.8.251.39:2181"

从10.8.251.37远程推送到（10.8.251.38，10.8.251.39）安装

cd /opt/dolphinscheduler  
./install.sh

启停服务

# 一键停止集群所有服务  
sh ./bin/stop-all.sh  
   
# 一键开启集群所有服务  
sh ./bin/start-all.sh  
   
# 启停 Master  
sh ./bin/dolphinscheduler-daemon.sh stop master-server  
sh ./bin/dolphinscheduler-daemon.sh start master-server  
   
# 启停 Worker  
sh ./bin/dolphinscheduler-daemon.sh start worker-server  
sh ./bin/dolphinscheduler-daemon.sh stop worker-server  
   
# 启停 Api  
sh ./bin/dolphinscheduler-daemon.sh start api-server  
sh ./bin/dolphinscheduler-daemon.sh stop api-server  
   
# 启停 Alert  
sh ./bin/dolphinscheduler-daemon.sh start alert-server  
sh ./bin/dolphinscheduler-daemon.sh stop alert-server

### 登录url

http://10.8.251.37:12345/dolphinscheduler/ui/view/login/index.html

### 用户名密码

| admin | admin@2022 |
| --- | --- |
| yfl | yfl@2022 |

# dolphinscheduler 单机环境部署

### mysql安装

rpm --import https://repo.mysql.com/RPM-GPG-KEY-mysql-2022  
mysql install:  
rpm -ivh http://mirrors.ustc.edu.cn/mysql-repo/mysql57-community-release-el7.rpm  
yum install mysql mysql-server

### 远程登录

取消mysql密码策略：  
set global validate\_password\_policy=0;  
set global validate\_password\_length=1;  
SET PASSWORD FOR 'root'@'localhost' = PASSWORD('123456');  
GRANT ALL PRIVILEGES ON \*.\* TO 'root'@'%' IDENTIFIED BY '123456'  
FLUSH PRIVILEGES;

### mysql启停

systemctl start/stop mysqld

### 用户名密码

root/123456

## zookeeper 安装

### 安装目录

/opt/apache-zookeeper-3.5.10-bin

### 配置文件修改

/opt/apache-zookeeper-3.5.10-bin/conf/zoo.cfg

dataDir=/opt/apache-zookeeper-3.5.10-bin/data

### 启动zookeeper

cd /opt/apache-zookeeper-3.5.10-bin/bin  
./zkServer.sh start

### 停止zookeeper

cd /opt/apache-zookeeper-3.5.10-bin/bin  
./zkServer.sh stop

## dolphinscheduler 环境准备

### 修改 /etc/hosts

10.8.251.230 master-server

### 配置用户免密

ssh-keygen -t rsa -P '' -f ~/.ssh/id\_rsa  
cat ~/.ssh/id\_rsa.pub >> ~/.ssh/authorized\_keys  
chmod 600 ~/.ssh/authorized\_keys

### 修改安装配置

/opt/apache-dolphinscheduler-2.0.5-bin/conf/config/install\_config.conf

# ---------------------------------------------------------  
# INSTALL MACHINE  
# ---------------------------------------------------------  
# 因为是在单节点上部署master、worker、API server，所以服务器的IP均为机器IP或者localhost  
ips="master-server"  
masters="master-server"  
workers="master-server:default"  
alertServer="master-server"  
apiServers="master-server"  
pythonGatewayServers="master-server"  
  
# DolphinScheduler安装路径，如果不存在会创建  
installPath="opt/dolphinscheduler"  
  
# 部署用户，填写在 \*\*配置用户免密及权限\*\* 中创建的用户  
deployUser="root"  
  
# ---------------------------------------------------------  
# DolphinScheduler ENV  
# ---------------------------------------------------------  
# JAVA\_HOME 的路径，是在 \*\*前置准备工作\*\* 安装的JDK中 JAVA\_HOME 所在的位置  
javaHome="/opt/jdk1.8.0\_341"  
  
# ---------------------------------------------------------  
# Database  
# ---------------------------------------------------------  
# 数据库的类型，用户名，密码，IP，端口，元数据库db。其中 DATABASE\_TYPE 目前支持 mysql, postgresql, H2  
# 请确保配置的值使用双引号引用，否则配置可能不生效  
DATABASE\_TYPE="mysql"  
SPRING\_DATASOURCE\_URL="jdbc:mysql://10.8.251.230:3306/dolphinscheduler?useUnicode=true&characterEncoding=UTF-8"  
SPRING\_DATASOURCE\_USERNAME="root"  
SPRING\_DATASOURCE\_PASSWORD="123456"  
  
# ---------------------------------------------------------  
# Registry Server  
# ---------------------------------------------------------  
# 注册中心地址，zookeeper服务的地址  
registryServers="localhost:2181"

### 安装dolphinscheduler

cd /opt/apache-dolphinscheduler-2.0.5-bin  
sh install.sh

数据库创建

CREATE DATABASE dolphinscheduler DEFAULT CHARACTER SET utf8 DEFAULT COLLATE utf8\_general\_ci;

刷新数据

sh script/create-dolphinscheduler.sh

### 启动 DolphinScheduler

cd /opt/dolphinscheduler/bin/  
启动服务：  
./start-all.sh   
停止服务  
./stop-all.sh

|  | 用户名 | 密码 |
| --- | --- | --- |
| 1 | admin | [admin@2022.com](mailto:admin@2022.com) |
| 2 | ylf | [yfl@2022.com](mailto:yfl@2022.com) |

# ELK集群安装

端口配置

| ip | 端口 |  |
| --- | --- | --- |
| 10.8.251.62 | 5601 | kibana |
| 10.8.251.62/10.8.251.63/10.8.251.64 | 9200 | es |
| 10.8.251.64 | 8888 | logstash |

| 机器名 | ip |  |
| --- | --- | --- |
| elk-server1 | 10.8.251.62 | es/kibana |
| elk-server2 | 10.8.251.63 | es |
| elk-server3 | 10.8.251.64 | es/logstash |

创建es用户

useradd es  
 passwd es 123456

创建目录

mkdir -p /home/es/data   
mkdir -p /home/es/logs

挂载目录

/dev/sda3 /home/es/data ext4 defaults 0 0

修改/etc/sysctl.conf文件

fs.file-max=65536  
vm.max\_map\_count=262144  
使用sysctl -p使条件生效

修改 /etc/security/limits.conf

\* soft nofile 65536  
\* hard nofile 65536  
\* soft nproc 65536  
\* hard nproc 65536  
\* soft memlock unlimited  
\* hard memlock unlimited

节点1配置文件修改

#节点名称  
node.name: elk-server1  
#数据目录  
path.data: /home/es/data  
#日志目录  
path.logs: /home/es/logs  
#本节点ip  
network.host: 0.0.0.0  
#端口  
http.port: 9200  
#集群主节点候选列表  
discovery.seed\_hosts: ["elk-server1", "elk-server2", "elk-server3"]  
#集群初始主结点列表  
cluster.initial\_master\_nodes: ["elk-server1","elk-server2","elk-server3"]  
#集群启动到2个节点之前，阻止数据恢复  
gateway.recover\_after\_nodes: 3  
#跨域访问设置  
http.cors.enabled: true  
http.cors.allow-origin: "\*"

节点2配置文件修改

#节点名称  
node.name: elk-server2  
#数据目录  
path.data: /home/es/data  
#日志目录  
path.logs: /home/es/logs  
#本节点ip  
network.host: 0.0.0.0  
#端口  
http.port: 9200  
#集群主节点候选列表  
discovery.seed\_hosts: ["elk-server1","elk-server2","elk-server3"]  
#集群初始主结点列表  
cluster.initial\_master\_nodes: ["elk-server1","elk-server2","elk-server3"]  
#集群启动到2个节点之前，阻止数据恢复  
gateway.recover\_after\_nodes: 3  
#跨域访问设置  
http.cors.enabled: true  
http.cors.allow-origin: "\*"  
#数据结点  
#node.master: false

节点3配置文件修改

#节点名称  
node.name: elk-server3  
#数据目录  
path.data: /home/es/data  
#日志目录  
path.logs: /home/es/logs  
#本节点ip  
network.host: 0.0.0.0  
#端口  
http.port: 9200  
#集群主节点候选列表  
discovery.seed\_hosts: ["elk-server1","elk-server2","elk-server3"]  
#集群初始主结点列表  
cluster.initial\_master\_nodes: ["elk-server1","elk-server2","elk-server3"]  
#集群启动到2个节点之前，阻止数据恢复  
gateway.recover\_after\_nodes: 3  
#跨域访问设置  
http.cors.enabled: true  
http.cors.allow-origin: "\*"

权限修改

chown -R es:es /home/es/elasticsearch-7.6.1  
 chown -R es:es /home/es/logs  
 chown -R es:es /home/es/data/

启动es

/home/es/elasticsearch-7.6.1/bin/elasticsearch -d

### Kibana安装配置

server.host: "0.0.0.0"  
server.port: 5601  
elasticsearch.hosts: ["http://10.8.251.62:9200","http://10.8.251.63:9200","http://10.8.251.64:9200"]  
kibana.index: ".kibana"

kibana启动

/opt/kibana-7.6.1-linux-x86\_64/bin/kibana --allow-root

命令

get \_cat/nodes?v

### logstash 配置log

input {  
 tcp {  
 #host => "0.0.0.0"  
 port => 8888  
 type => "from\_log4"  
 }  
}  
output {  
 if [type]=="from\_log4"{  
 elasticsearch {  
 hosts => ["10.8.251.62:9200","10.8.251.63:9200","10.8.251.64:9200"]  
 index => "log4"  
 user => "es"  
 password => "123456"  
 }  
 }  
}

### java项目中的日志文件logback-spring.xml中添加

<?xml version="1.0" encoding="UTF-8"?>  
<configuration>  
 <include resource="org/springframework/boot/logging/logback/base.xml" />  
 <appender name="LOGSTASH"  
 class="net.logstash.logback.appender.LogstashTcpSocketAppender">  
 <!--配置logStash 服务地址 -->  
 <destination>10.8.251.64:8888</destination>  
 <!-- 日志输出编码 -->  
 <encoder charset="UTF-8"  
 class="net.logstash.logback.encoder.LoggingEventCompositeJsonEncoder">  
 <providers>  
 <timestamp>  
 <timeZone>UTC</timeZone>  
 </timestamp>  
 <pattern>  
 <pattern>  
 {  
 "logLevel": "%level",  
 "serviceName": "${springAppName:-}",  
 "pid": "${PID:-}",  
 "thread": "%thread",  
 "class": "%logger{40}",  
 "rest": "%message"  
 }  
 </pattern>  
 </pattern>  
 </providers>  
 </encoder>  
 </appender>  
  
 <root level="DEBUG">  
 <appender-ref ref="LOGSTASH" />  
 <appender-ref ref="CONSOLE" />  
 </root>  
</configuration>

pom.xml 添加

<dependency>  
 <groupId>net.logstash.logback</groupId>  
 <artifactId>logstash-logback-encoder</artifactId>  
 <version>6.6</version>  
 </dependency>

### logstash 启动

nohup ./logstash -f /opt/logstash-7.6.1/config/log4.conf &

# ELK单机安装

/home/es/elasticsearch-7.6.1/config/elasticsearch.yml

#集群名称  
cluster.name: yfl-es-cluster  
  
#节点名称  
node.name: elk-server1  
#数据目录  
path.data: /home/es/data  
#日志目录  
path.logs: /home/es/logs  
#本节点ip  
network.host: 0.0.0.0  
#端口  
http.port: 9200  
#集群主节点候选列表  
cluster.initial\_master\_nodes: ["elk-server1"]  
#跨域访问设置  
http.cors.enabled: true  
http.cors.allow-origin: "\*"

### 启动单机es

su - es  
cd /home/es/elasticsearch-7.6.1/bin  
./elasticsearch -d

## kabana 配置

/opt/kibana-7.6.1-linux-x86\_64/config/kibana.yml

server.host: "0.0.0.0"  
server.port: 5601  
elasticsearch.hosts: ["http://10.8.251.62:9200"]  
kibana.index: ".kibana"

### kibana启动

nohup /opt/kibana-7.6.1-linux-x86\_64/bin/kibana --allow-root &

## Log stash 配置

/opt/logstash-7.6.1/config/log4.conf

input {  
 tcp {  
 #host => "0.0.0.0"  
 port => 8888  
 type => "from\_log4"  
 }  
}  
output {  
 if [type]=="from\_log4"{  
 elasticsearch {  
 hosts => ["10.8.251.62:9200"]  
 index => "log4"  
 user => "es"  
 password => "123456"  
 }  
 }  
}  
~

### logstash启动

cd /opt/logstash-7.6.1/bin  
nohup ./logstash -f /opt/logstash-7.6.1/config/log4.conf &

# Mysql8 安装

### 下载rpm包

wget https://dev.mysql.com/get/Downloads/mysql-8.0.27-1.el7.x86\_64.rpm-bundle.tar

### 解压

tar -xvf mysql-8.0.27-1.el7.x86\_64.rpm-bundle.tar

### 安装

rpm -ivh mysql-community-common-8.0.27-1.el7.x86\_64.rpm   
 rpm -ivh mysql-community-client-plugins-8.0.27-1.el7.x86\_64.rpm   
 rpm -ivh mysql-community-libs-8.0.27-1.el7.x86\_64.rpm   
 rpm -ivh mysql-community-client-8.0.27-1.el7.x86\_64.rpm  
 rpm -ivh mysql-community-server-8.0.27-1.el7.x86\_64.rpm

### mysql 目录权限

chown -R mysql:mysql /var/lib/mysql/

### 初始化数据库

添加忽略大小写配置

mysqld --initialize --lower-case-table-names=1

### 修改配置文件 /etc/my.cnf

必须与初始化相同配置，否则启动报错

[mysqld]  
lower\_case\_table\_names=1

### 修改密码

alter USER 'root'@'localhost' IDENTIFIED BY 'CloudGence1019';

### 修改权限

update user set host = "%" where user='root';  
select host, user, authentication\_string, plugin from user;  
flush privileges;

# doris备份脚本

### datax执行机器

10.8.251.37

### 执行脚步目录

/opt/sh

### 同步表列表

/opt/sh/table\_list.data

同步脚步执行json

{  
"job": {  
 "content": [  
 {  
 "reader":{  
 "name":"mysqlreader",  
 "parameter":{  
 "column":["\*"],  
 "connection":[  
 {  
 "jdbcUrl":[  
 "jdbc:mysql://10.8.251.57:9030/EIFINI\_BCS"  
 ],  
 "table":[  
 "${readTb}"  
 ]  
 }  
 ],  
 "password":"Bcs221018",  
 "username":"bcs"  
 }  
 },  
 "writer": {  
 "name": "starrockswriter",  
 "parameter": {  
 "username": "bcs",  
 "password": "Bcs221018",  
 "database": "EIFINI\_BCS",  
 "table": "${writeTb}",  
 "column":["\*"],  
 "presql": ["truncate table ${writeTb}"],  
 "maxBatchRows": "100000",  
 "flushInterval": "1000000",  
 "jdbcUrl": "jdbc:mysql://10.8.251.58:9030/",  
 "loadUrl": ["10.8.251.58:8030"],  
 "loadProps": {}  
 }  
 }   
  
 }  
 ],  
 "setting": {  
 "speed": {  
 "channel": "10"  
 }  
 }  
 }  
}

### Shell 同步脚本

#!/bin/bash  
  
# 脚本所在目录及脚本名称  
script\_dir=$( cd "$( dirname "$0" )" && pwd )  
# script\_name=$(basename ${0})  
if [ -z "$1" ]; then  
 exec < ${script\_dir}/table\_list.data  
 while read table\_name; do  
 echo "正在同步表：${table\_name} ......"  
 python /opt/soft/datax/bin/datax.py ${script\_dir}/comm.json -p "-DreadTb=$table\_name -DwriteTb=$table\_name" --jvm="-Xms8G -Xmx8G"  
 done  
else  
 table\_name=$1  
 python /opt/soft/datax/bin/datax.py ${script\_dir}/comm.json -p "-DreadTb=$table\_name -DwriteTb=$table\_name" --jvm="-Xms8G -Xmx8G"  
fi

### 脚步执行命令

cd /opt/sh  
./run.sh 同步执行/opt/sh/table\_list.data 中所有的表  
./run.sh table\_name 单表同步