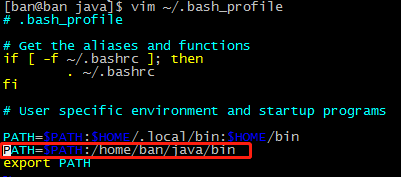
# Java部署

Zookeeper 安装包：java8.tar.gz

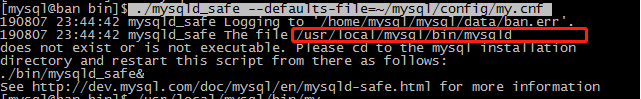
1. 解压文件：
   1. tar -zxvf ./java8.tar.gz -C java
2. 切换到本地
   1. mv ./java ~
3. 配置环境变量
   1. vim ~/.bash\_profile
   2. 
   3. PATH=$PATH:/home/ban/java/bin
4. 加载生效
   1. source ~/.bash\_profile

# Mysql部署

安装包：mysql-5.6.35-linux-glibc2.5-x86\_64.tar.gz

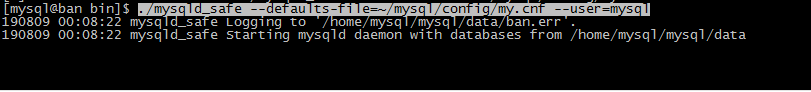
1. 创建目录
   1. mkdir mysql
2. 解压文件夹
   1. tar -zxvf ./mysql-5.6.35-linux-glibc2.5-x86\_64.tar.gz -C mysql
3. 使用前需要给mysql创建用户和密码
   1. 目的：
      1. 安全性：隔离性两个用户，防止出现漏洞被人攻破权限，就不能随便访问其他用户的资料了，只能访问mysql用户的数据。也能防止主用户随便修改mysql用户的数据。
      2. 影响隔离：防止出个bug，影响到机器性能的时候，使得主账户的用户拖垮性能，但前提是有进行资源分配（像CPU、内存之类每个用户都有进行限制带下）
   2. 操作：先登录root用户
      1. 创建mysql用户组
         1. groupadd mysql
      2. 创建用户并指定用户组：useradd -g mysql mysql
         1. -g :指定用户组
      3. 把mysql转移到mysql用户
         1. [ban@ban ~]$ scp -r ./mysql/ mysql@localhost:~
      4. 配置文件配置
         1. 创建配置文件夹
            1. mkdir ./mysql/config
         2. [ban@ban ~]$ cp ./mysql/support-files/my-default.cnf ./mysql/config/my.cnf
         3. 修改配置文件

|  |
| --- |
| # These are commonly set, remove the # and set as required.  basedir = /home/mysql/mysql  datadir = /home/mysql/mysql/data  port = 3306  log-err = /home/mysql/mysql/data/error.log  pid-file = /home/mysql/mysql/data/mysql.pid  # server\_id = .....  # socket = ..... |

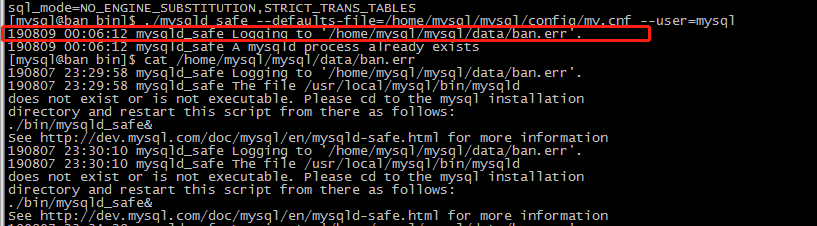
* + 1. 安装依赖库autoconf，否则不能启动
       1. yum -y install autoconf
    2. 安装依赖库libaio，否则不能启动
       1. yum install libaio\*
    3. 初始化数据（还不是启动程序）
       1. 必须切换到程序根目录，否则会报脚本找不到的情况
          1. cd mysql/
       2. 初始化数据
          1. ./scripts/mysql\_install\_db --verbose --user=mysql --defaults-file=~/mysql/config/my.cnf
    4. 启动进程
       1. 启动命令
          1. ./mysqld\_safe --defaults-file=~/mysql/config/my.cnf --user=mysql
       2. 报错：
          1. 
          2. 该文件在安装包的mysql/bin/mysqld，但是脚本mysqld\_safe默认查的是/usr/local/mysql/bin/mysqld，所以需要手动修改
       3. 登录root，软链接修改（mysqld\_safe脚本启动会访问该路径）
          1. [root@ban bin]# ln -s /home/mysql/mysql/ /usr/local/mysql
          2. 注意：也可以不用这个方案：可以修改mysql\_safe脚本里面的路径达到访问到mysql下的路径。

参考资料：<https://www.cnblogs.com/moss_tan_jun/p/6080058.html>

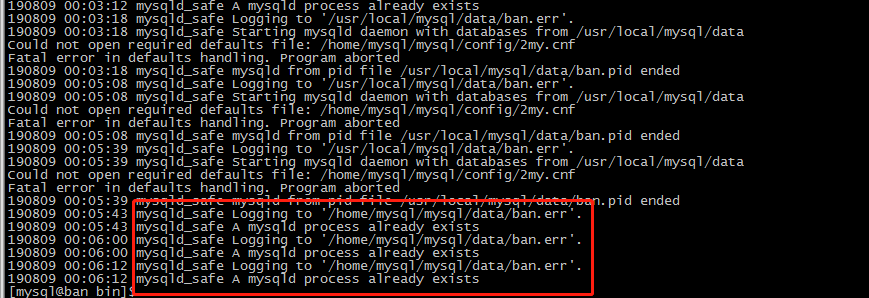
* + - 1. ~~修该用户组，使mysql用户可以访问（暂时不需要）~~
         1. ~~cd /usr/local/~~
         2. ~~[root@ban local]# chown -R mysql ./mysql/~~
      2. 重新启动程序
         1. ./mysqld\_safe --defaults-file=~/mysql/config/my.cnf --user=mysql
         2. 启动成功：启动后如果光标闪烁才代表启动成功。

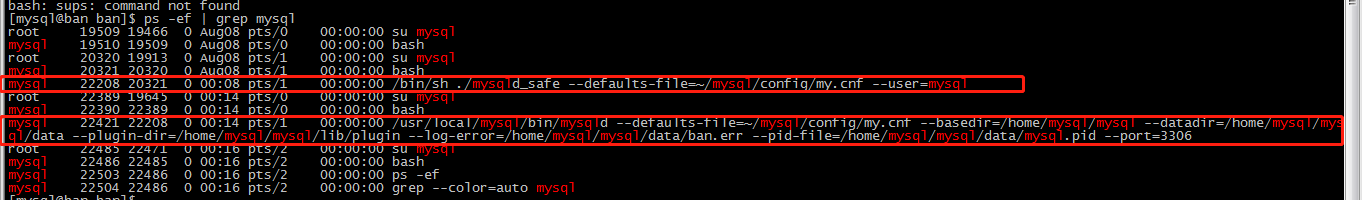


* + - * 1. 启动失败：如果执行失败会有提示信息



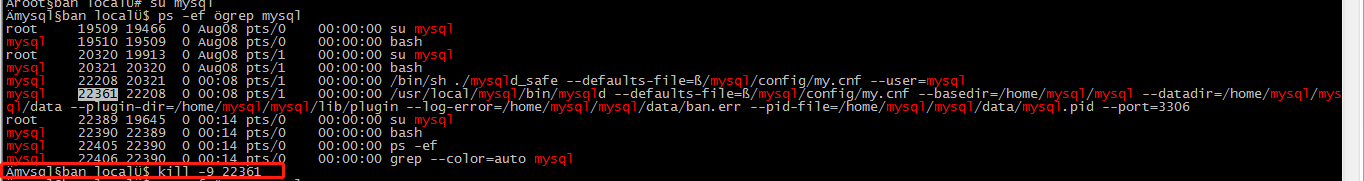
日志信息



* + 1. 查看进程是否启动
       1. [mysql@ban ban]$ ps -ef | grep mysql
          1. 
       2. 发现有两个进程：mysqld\_safe、mysqld
       3. 区别（列举其他其中一个区别，具体查看下面的文章）：
          1. mysqld\_safe相当于守护进程，守护mysqld，如果mysqld挂了，会把mysqld重新启动
       4. 其他区别：

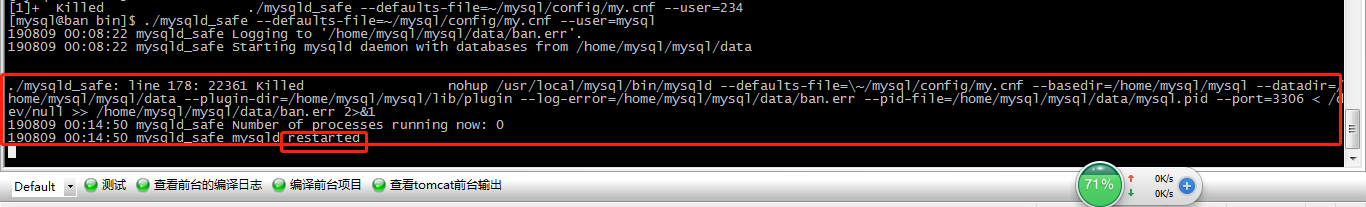
|  |
| --- |
| 1. 检查系统和选项。  2. 检查MyISAM表。  3. 保持MySQL服务器窗口。  4. 启动并监视mysqld，如果因错误终止则重启。  5. 将mysqld的错误消息发送到数据目录中的host\_name.err 文件。  6. 将mysqld\_safe的屏幕输出发送到数据目录中的host\_name.safe文件。 |

* + - 1. 验证
         1. 杀死mysqld进程

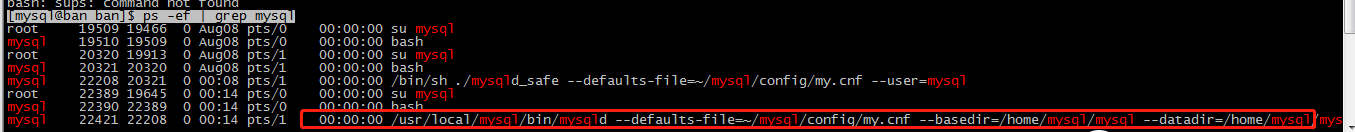


* + - * 1. 检查是否重启

查询守护进程的日志（重启日志）



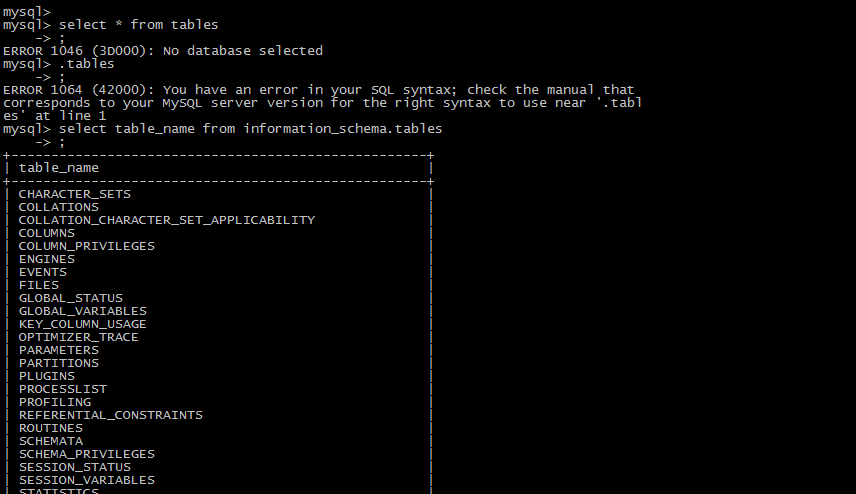
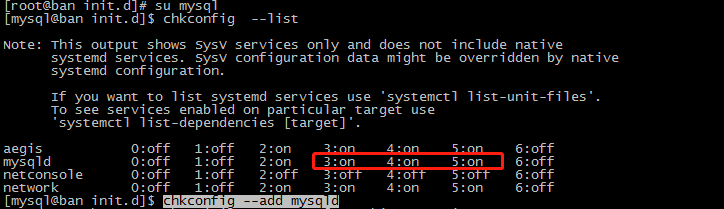
[mysql@ban ban]$ ps -ef | grep mysql



* + - * 1. 参考资料

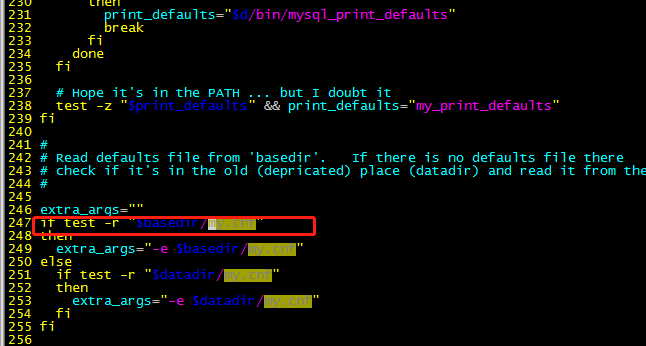
mysqld\_safe与mysqld区别详解

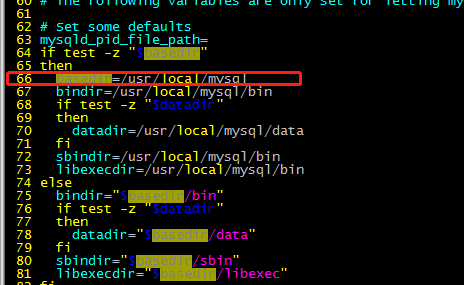
<https://www.cnblogs.com/shamo89/p/10164089.html>

* + 1. 登录操作：
       1. ./mysql -uroot -p （root账户密码默认为空）
       2. 查询所有表
          1. select table\_name from information\_schema.table;
          2. 
    2. 配置开机自动启动进程
       1. root权限下：复制脚本
          1. cp /home/mysql/mysql/support-files/mysql.server /etc/rc.d/init.d/mysqld
       2. 添加为服务
          1. chkconfig --add mysqld
       3. 查看服务列表
          1. chkconfig --list
          2. 
       4. 检查是否生效
          1. 开启为生效状态：**chkconfig --level 345 mysqld on**
          2. 看到3、4、5状态为开或者为 on 则表示成功。如果是 关或者 off 则执行一下：**chkconfig --level 345 mysqld on**
       5. 模拟启动
          1. service mysqld start
          2. 报错：无法生成日志文件



* + - * 1. 查看/home/mysql/mysql/support-files/mysql.server



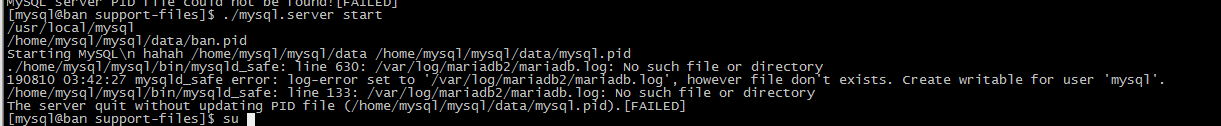


可知道查找默认的配置文件在/usr/local/mysql/my.cnf，所以要把执行一下语句

cp /usr/local/mysql/config/my.cnf /usr/local/mysql/

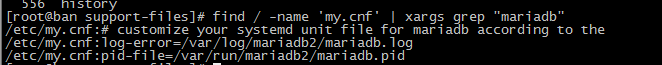
* + - * 1. 检查开机命令是否可用

./mysql.server start



查询哪个配置文件关联这个关键字mariadb

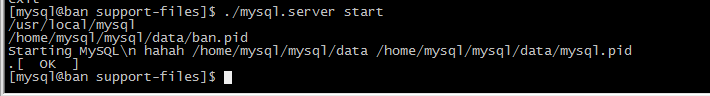
find / -name 'my.cnf' | xargs grep "mariadb"



解决这个问题登录root账户，因为这个/etc/my.cnf，才会有这个文件吗：mariadb，修改/etc/my.cnf文件吗即可。

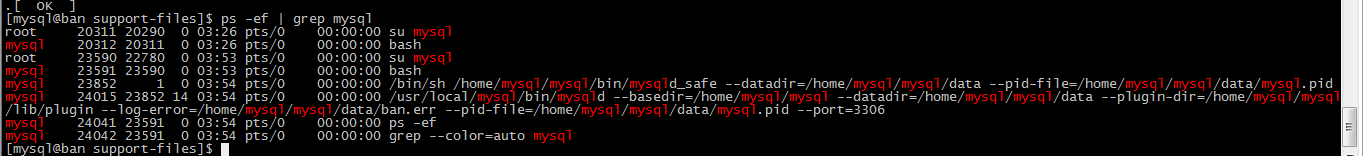
[root@ban support-files]# mv /etc/my.cnf /etc/my.cnf.bak

再次执行./mysql.server start



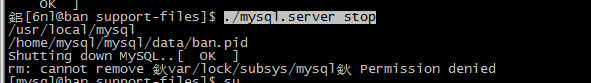
查看进程是否启动

ps -ef | grep mysql



* + - * 1. 检查是否可用停止

./mysql.server stop

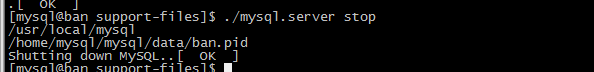


该文件路径由mysql.server脚本里面指定: lockdir='/var/lock/subsys'。权限不足，因为该文件夹是root账户创建，必须赋予mysql用户组

解决：

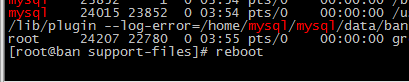
[root@ban support-files]# chown mysql:mysql /var/lock/subsys/mysql

mysql用户再次登录./mysql.server stop

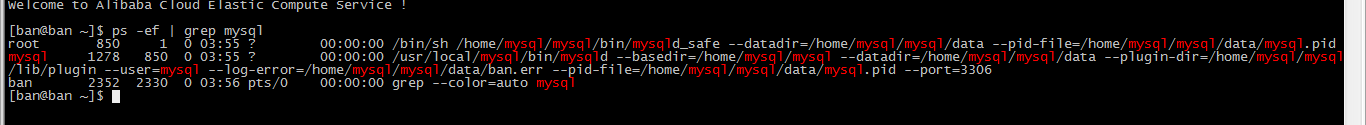


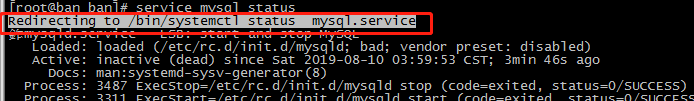
* + - * 1. 验证重启是否会自动开启服务

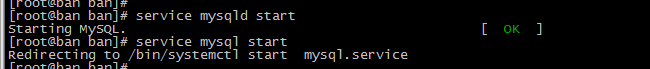
reboot



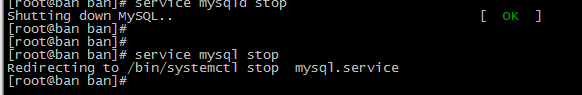
[ban@ban ~]$ ps -ef | grep mysql



* + - 1. 命令的区别：service mysql xx和service mysqld xx效果一样，因为service mysql的命令会重定向到Redirecting to /bin/systemctl status mysql.service
         1. 
      2. 启动命令
         1. service mysqld start 或者service mysql start 效果一样

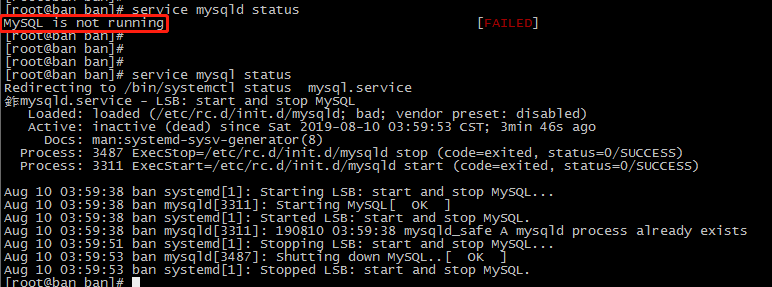


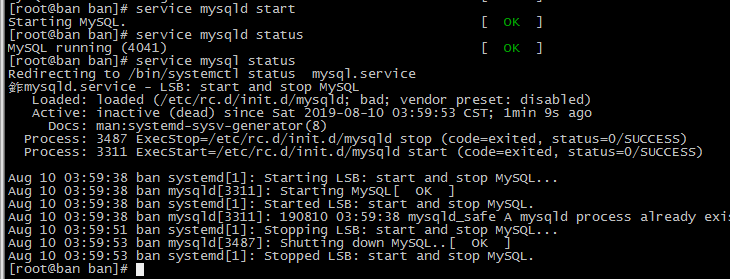
* + - 1. 停止命令
         1. service mysqld stop 或者service mysql stop 效果一样



* + - 1. 状态命令
         1. service mysqld status或者service mysql status效果一样

关闭的情况下



开启的情况下

# zookeeper部署

Zookeeper 安装包：zookeeper-3.4.6.tar.gz

## 解压安装包

* 1. [ban@ban soft]$ mkdir zookeeper
  2. [ban@ban soft]$ tar -zxvf ./zookeeper-3.4.6.tar.gz -C zookeeper
  3. [ban@ban soft]$ mv ./zookeeper ~/

## 修改配置文件

* 1. 拷贝配置文件模版
     1. [ban@ban ~]$ cp ~/zookeeper/conf/zoo\_sample.cfg ~/zookeeper/conf/zoo.cfg
  2. 修改配置：[ban@ban ~]$ vim ~/zookeeper/conf/zoo.cfg

|  |
| --- |
| #zk心跳间隔，单位是毫秒  tickTime=2000  #1、这个配置项是用来配置 Zookeeper 接受客户端初始化连接时最长能忍受多少个心跳时间间隔，当已经超过 10 个心跳的时间也就是（ticktime）长度后 Zookeeper 服务器还没有收到客户端的返回信息，那么表明这个客户端连接失败，总的时间长度就是：10\*2000 = 20s  #就是说：Follower连接和同步Leader的超时时间(10 tickTime)  initLimit=10  #这个配置项表示 Leader 与 Follower 之间发送消息，请求和应答时间长度，最长不能超过多少个 tickTime 的时间长度，总的时间长度就是：5\*2000 = 10s  #即：#Follower同步Leader的超时时间(5 tickTime)  syncLimit=5  #存储内存中数据库快照的位置，顾名思义就是 Zookeeper 保存数据的目录，默认情况下，，如果不指定日志目录的话Zookeeper 将写数据的日志文件也保存到这个目录里  #数据文件存放路径.，需要提前创建（必须使用绝对路径，相对路径会报错）  dataDir=/home/ban/data/zookeeper  #日志存放路径，如果不配置使用dataDir，需要提前创建（必须使用绝对路径，相对路径会报错）  dataLogDir=/home/ban/datalog/zookeeper  # the port at which the clients will connect  #2181：对外提供端口  clientPort=2181  #设置保留多少个snapshot,之前的则删除  #autopurge.snapRetainCount=3  #24个小时清理一次多余的日志  #表示会自动清除24小时以前的历史数据(如果业务非常繁忙,可能会出现忙时清理数据的问题,此时可修改参数=0,然后配置crontab任务对日志进行手动处理)  autopurge.purgeInterval=24  #server.A = B:C:D  #A：表示这是第几号服务器  #B：服务器的 IP 地址  #C：服务器与集群中的 Leader 服务器交换信息的端口  #D：一旦集群中的 Leader 服务器挂了，需要一个端口重新进行选举，选出一个新的 Leader  #2281：内部同步端口  #2381：节点挂了，选举端口  server.1=172.16.239.107:2281:2381 |

* 1. 创建对应的dataDir数据文件夹和dataLogDir日志文件夹
     1. [ban@ban ~]$ mkdir -p ~/data/zookeeper
     2. [ban@ban ~]$ mkdir -p ~/datalog/zookeeper

## 创建myid文件

* 1. 按server.后面跟的数字在相应ip地址的dataDir目录下创建myid文件,文件内容 与前面server.后面跟的数字一致.
     1. 比如server.1=172.16.239.107:2281:2381，在172.16.239.107机器的dataDir目录下创建myid文件，文件内容为1
  2. 创建myid
     1. [ban@ban ~]$ vim ~/data/zookeeper/myid
     2. 

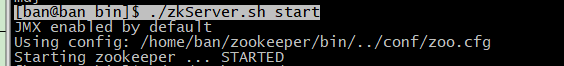
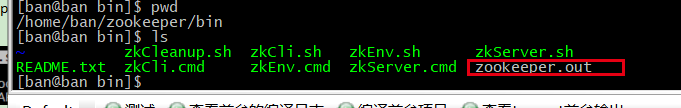
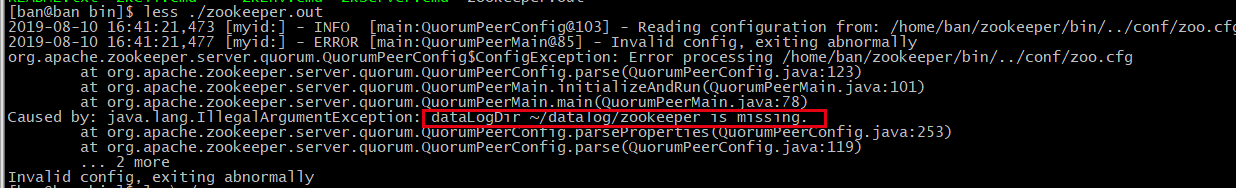
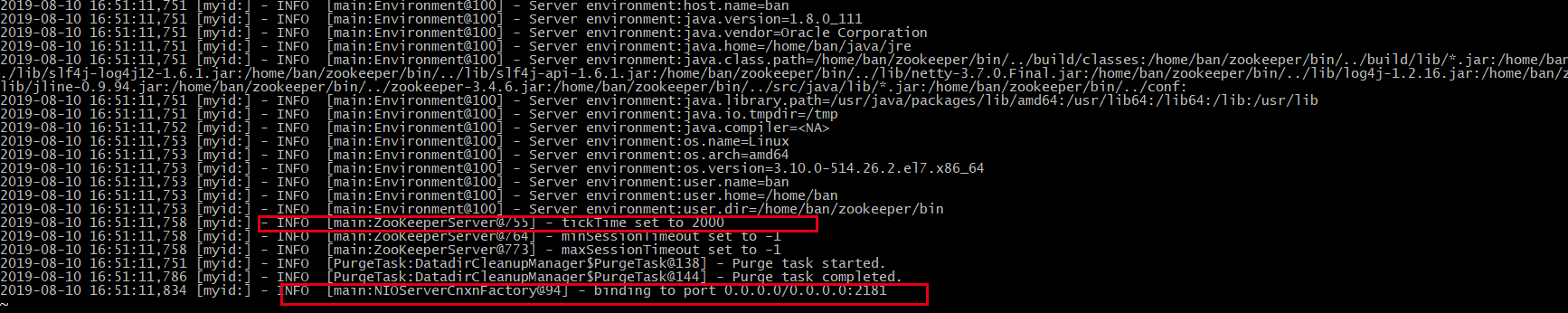
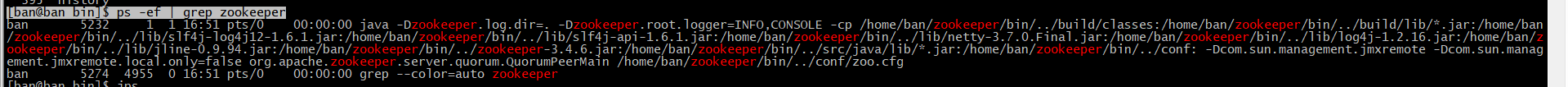
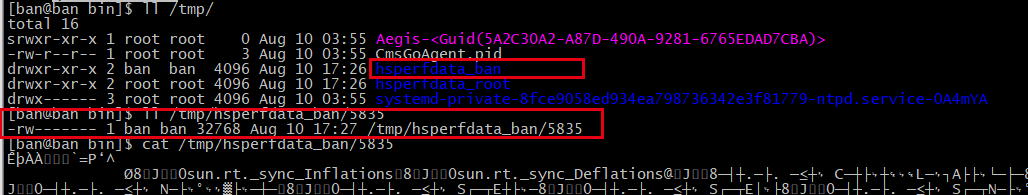
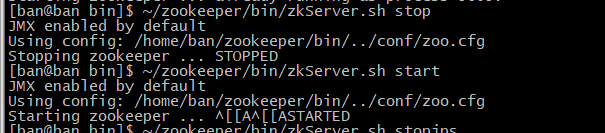
## 配置日志存储路径

根据环境修改各日志存储路径。可以不做修改

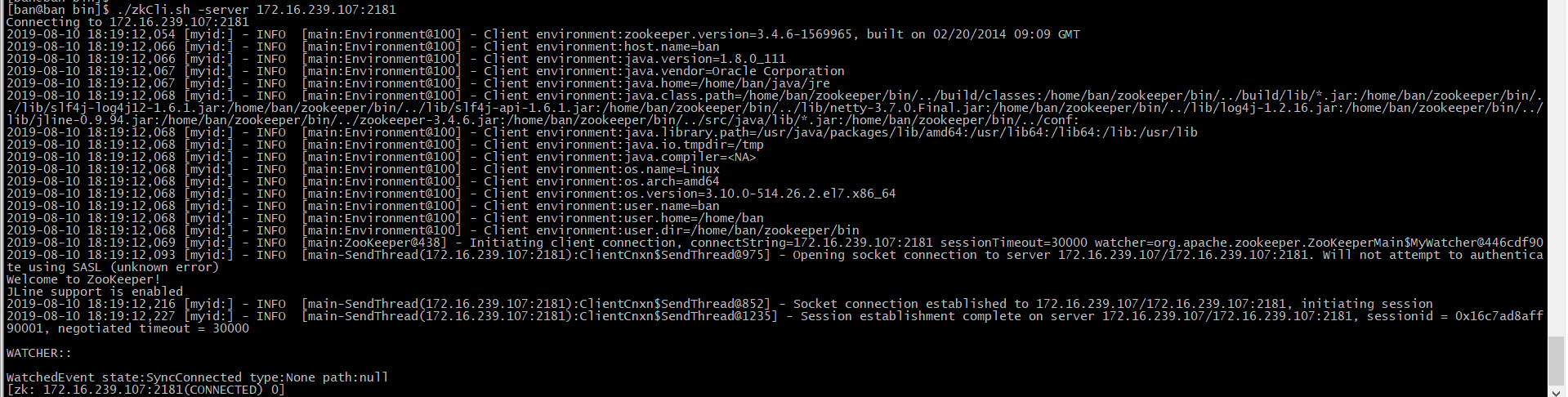
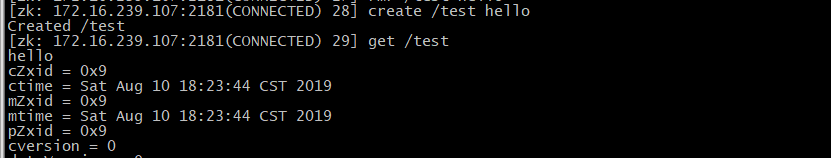
* 1. [ban@ban conf]$ cat ~/zookeeper/conf/log4j.properties

|  |
| --- |
| # Define some default values that can be overridden by system properties  zookeeper.root.logger=INFO, CONSOLE  zookeeper.console.threshold=INFO  zookeeper.log.dir=.  zookeeper.log.file=zookeeper.log  zookeeper.log.threshold=DEBUG  zookeeper.tracelog.dir=.  zookeeper.tracelog.file=zookeeper\_trace.log  #  # ZooKeeper Logging Configuration  #  # Format is "<default threshold> (, <appender>)+  # DEFAULT: console appender only  log4j.rootLogger=${zookeeper.root.logger} |

## 启动zk服务

* 1. [ban@ban bin]$ cd /home/ban/zookeeper/bin
  2. [ban@ban bin]$ ./zkServer.sh start
     1. 
  3. 检查服务是否开启
     1. 
  4. 没有开启检查日志
     1. 
     2. 检查原因
        1. 
        2. 找不到对应的路径，语法错误，表明不能使用相对路径，必须使用绝对路径，重新配置即可。
  5. 重新启动
     1. [ban@ban bin]$ ./zkServer.sh start
     2. 查询日志
        1. [ban@ban bin]$ less ./zookeeper.out
        2. 
     3. 查询进程
        1. [ban@ban bin]$ ps -ef | grep zookeeper
        2. 
     4. 检查jps
        1. Image
        2. 检查 /tmp/
           1. 
           2. 发现当前是发现当前用户组是 ban，但是该文件可能赋值为root或mysql用户组的权限，导致当前进程根本就没有权限写进去嘛
        3. 使用root修改权限[root@ban ~]# chown ban:ban /tmp/hsperfdata\_ban/
        4. 重启启动zookeeper进程（成功）
           1. 
           2. 
     5. 查看启动参数
        1. ./zkServer.sh print-cmd

## 连接测试

* 1. ./zkCli.sh -server 172.16.239.107:2181
     1. 
  2. 测试并写入信息
     1. 格式：create 写入的目录，以根目录开头 写入的内容
        1. create /test hello
     2. 创建并查看
        1. [zk: 172.16.239.107:2181(CONNECTED) 28] create /test hello
        2. Created /test
        3. [zk: 172.16.239.107:2181(CONNECTED) 29] get /test
        4. 

# kafka部署

Kafka安装包：kafka\_2.11-0.8.2.2.tar.gz

Broker规划：两个broker

172.16.239.107

Broker1 Prot：9001

Broker1 Prot：9002

## 解压安装包

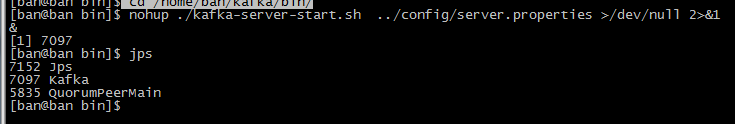
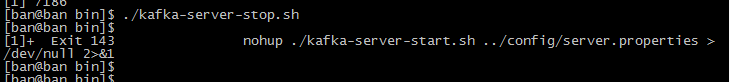
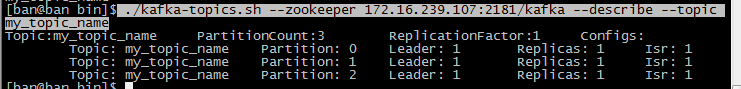
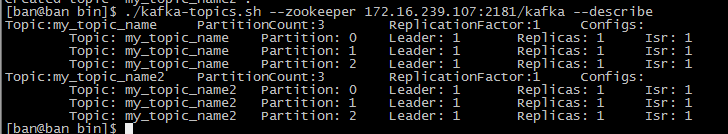
1. [ban@ban soft]$ tar -zxvf ./kafka-2.10.tar.gz
2. [ban@ban soft]$ mv ./kafka ~/

## 配置文件修改（根据实际环境修改）

1. 消费者配置文件修改：consumer.properties
   1. 修改zk连接地址，并指定zk存储数据目录（集群的话要指定多个）：zookeeper.connect=172.16.239.107:2181/kafka
   2. 
2. 生产者配置文件修改：producer.properties（集群需要指定多个）
   1. metadata.broker.list=172.16.239.107:9001,172.16.239.107:9002 （使用多个broker的端口）
3. 修改服务端配置，如果集群需要配置多台机器：server.properties

|  |
| --- |
| ############################# Server基本信息 #############################  #Broker的ID，集群中的每个Server需要唯一（不同的borker需要修改）  broker.id=1  #是否自动创建队列（建议为false，防止测试时程序创建过多无用的topic，影响线上故障排查）  auto.create.topics.enable=false  ############################# Socket Server配置 #############################  #Kafka监听端口（如果集群在一个集群需要修改）  port=9001  #Kafka绑定的IP地址  host.name=172.16.239.107  #处理网络请求的network threads数量 如果是多台集群数量是集群机器-1  num.network.threads=3    #处理网络请求的IO threads数量  num.io.threads=16  #SocketServer使用的发送Buffer(SO\_SNDBUF) 1048576byte==1M  socket.send.buffer.bytes=1048576  #SocketServer使用的接收Buffer(SO\_RCVBUF) 1048576byte==1M  socket.receive.buffer.bytes=1048576  #SocketServer可以接收的最大请求数(防止OOM) 104857600byte=100M  socket.request.max.bytes=104857600  ############################# Log基本信息 #############################  #用于存放Kafka的数据（如果集群在一个集群需要修改）  log.dirs=/home/ban/hdata/kafka2\_1  #每个Topic的日志分区默认数量  num.partitions=3  #Broker自动平衡  auto.leader.rebalance.enable=true  ############################# Log刷新策略 #############################  #接收到的消息数量后，会立即flush数据到磁盘中  #log.flush.interval.messages=10000  #多长时间后，会flush数据到磁盘中(ms) 1秒  #log.flush.interval.ms=1000  ############################# Log保留策略 #############################  #日志文件最大生存时间(22小时)  log.retention.hours=22  #当日志总大小超过该大小时会修剪 (1G)  #log.retention.bytes=1073741824  #日志文件的最大大小(1G),当大小超过该值时，会创建一个新的Log Segment  log.segment.bytes=1073741824  #检查日志文件的间隔时间(ms), 5分钟（检查日志分段文件的间隔时间,以确定是否文件属性是否到达删除上面两个要求（距离现在的时间，占用大小），是否超过22个小时，是否超过1G大小）  log.retention.check.interval.ms= 300000  #是否开启日志压缩  log.cleaner.enable=false  ############################# Zookeeper #############################  #Zookeeper连接URL串, 若使用了chroot，需要提前创建目录  zookeeper.connect= 172.16.239.107:2181/kafka  #ZooKeeper的连接超时时间(ms)  zookeeper.connection.timeout.ms=6000 |

## 脚本启动测试

1. 启动注意事项：
   1. 注：kafka依赖于zookeeper，所以kafka启动之前，一定要先启动zookeeper
2. 启动命令（建议做成脚本）
   1. cd /home/ban/kafka/bin/
   2. 后台模式启动：nohup ./kafka-server-start.sh ../config/server.properties >/dev/null 2>&1 &
   3. 检查是否启动：jps
      1. 
3. 停止命令（建议做成脚本）
   1. cd /home/ban/kafka/bin/
   2. ./kafka-server-stop.sh
   3. 
4. 测试：
   1. 创建一个topic（Kafka 2.2 开始的版本建议替换参数zookeeper 为bootstrap-server，地址为kafka的连接地址和端口）（参数:partitions、replication-factor是必填项）
      1. Kafka 2.2 之前的版本 与Kafka 2.2 开始的版本最大的区别就是参数从--zookeeper 改成--bootstrap-server，也就是从连接zookeeper改成连接到kafka。Kafka 2.2 开始的版本既可以连kafka也可以连zookeeper，官方推荐使用连接kafka。
         1. 避免可以利用zookeeper绕过kafka的安全体系，从而每个用户都可以看到所有topic的信息，没有进行权限过滤（指定某些topic信息可以给某个用户查询）
         2. 减少zookeeper依赖，不需要同时维护zookeeper和kafka的连接信息
      2. Kafka 2.2 之前的版本 replication-factor必须大于等于broker的数量
         1. bin/kafka-topics.sh --zookeeper 172.16.239.107:2181/kafka --create --topic my\_topic\_name --partitions 3 --replication-factor 3
      3. Kafka 2.2 开始的版本 replication-factor必须大于等于broker的数量
         1. bin/kafka-topics.sh --bootstrap-server broker\_host:port --create --topic my\_topic\_name --partitions 1 --replication-factor 1
         2. bin/kafka-topics.sh --bootstrap-server 172.16.239.107:9092 --create --topic my\_topic\_name --partitions 3 --replication-factor 3
      4. 创建成功
         1. 
   2. 查询所有topic列表（Kafka 2.2 开始的版本建议替换参数zookeeper 为bootstrap-server，地址为kafka的连接地址和端口）
      1. Kafka 2.2 之前的版本
         1. ./kafka-topics.sh --zookeeper 172.16.239.107:2181/kafka --list
         2. 
      2. Kafka 2.2 开始的版本
         1. ./kafka-topics.sh --bootstrap-server 172.16.239.107:9092 --create --topic my\_topicname --partitions 2 --replication-factor 1
         2. 
         3. 提示建议不能同时使用“\_”和“，”,但实际已经创建成功，只是建议不要这样创建
   3. 查询单个或所有topic的详细数据（Kafka 2.2 开始的版本建议替换参数zookeeper 为bootstrap-server，地址为kafka的连接地址和端口）
      1. 查询单个 ./kafka-topics.sh --zookeeper 172.16.239.107:2181/kafka --describe --topic my\_topic\_name
         1. 
      2. 查询所有topic的详细信息：./kafka-topics.sh --zookeeper 172.16.239.107:2181/kafka --describe
         1. 

# redis部署

## 文件准备

1. redis安装包：redis-3.0.6.tar.gz
2. ruby（脚本语言）：ruby-2.6.3.tar.gz（最新版截止20190815）
3. redis的ruby集群脚本依赖（java的jar包）：redis-4.1.2.gem（最新版截止20190815）
4. zlib库：zlib-1.2.11.tar.gz

## 节点预留（由于只有一台机器，只弄成伪集群）

172.16.239.107 部署：master-172.16.239.107:7000 , slave-172.16.239.107:7010

172.16.239.107 部署：master-172.16.239.107:7100 , slave-172.16.239.107:7110

172.16.239.107 部署：master-172.16.239.107:7200 , slave-172.16.239.107:7210

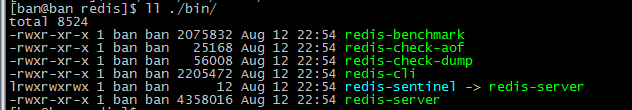
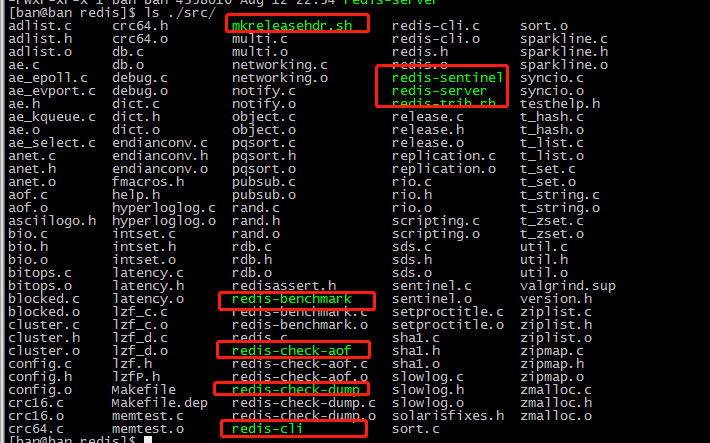
## 主机列表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **业务系统归属** | **主机名** | **Active IP** | **用户名** | **密码** |
| Redis服务 | ban | 172.16.239.107 | ban |  |
| Redis服务 |  |  |  |  |
| Redis服务 |  |  |  |  |

## 解压安装包

1. [ban@ban soft]$ tar -zxvf ./kafka-2.10.tar.gz
2. [ban@ban soft]$ mv ./kafka ~/

## 编译安装包

1. [ban@ban ~]$ cd /home/ban/redis/
2. make PREFIX=/home/ban/redis/ install
3. 可以看到redis根目录下生成bin文件夹，src文件夹下生成脚本
   1. bin文件夹
      1. 
   2. src文件夹生成脚本
      1. 

## 创建集群

1. 创建集群目录

|  |
| --- |
| mkdir ~/redis\_cluster  cd redis\_cluster  mkdir -p ~/redis\_cluster/7000/data  mkdir -p ~/redis\_cluster/7010/data  （PS：以端口命名节点目录，方便识别） |

1. 进入redis\_cluster目录，复制配置文件

|  |
| --- |
| [ban@ban ~]$ cp ~/redis/redis.conf ~/redis\_cluster/7000/redis\_7000.conf  [ban@ban ~]$ cp ~/redis/redis.conf ~/redis\_cluster/7010/redis\_7010.conf |

1. 修改配置文件（各个集群依次处理）

|  |
| --- |
| #redis\_70\*0.conf内容修改同上原理  vim ~/redis\_cluster/7000/redis\_7000.conf  ################修改conf文件的以下参数： #################  #端口（根据端口名修改）  port 7000  #绑定当前机器 IP（根据集群的机器修改）  bind 172.20.35.28  #修改磁盘上保存数据库文件的位置，可以默认不改:  dbfilename dump.rdb  #修改日志级别  #如果只要输出少量日志的话，可以用waring，可以默认不改  loglevel notice  #修改日志文件的位置（方便检查运行出错）（根据端口名修改,方便查找日志）  logfile "redis-log-7000.log”  #后台运行  daemonize yes  #集群模式  cluster-enabled yes  #集群配置文件,它是集群节点自动维护的文件，主要用于记录集群中有哪些节点、他们的状态以及一些持久化参数等，方便在重启时恢复这些状态（根据端口修改）  cluster-config-file nodes\_7000.conf  #集群中的节点能够失联的最大时间，超过这个时间，该节点就会被认为故障，15000毫秒，即15秒  cluster-node-timeout 15000  #pid文件（根据端口名修改）  pidfile redis\_7000.pid  #备份数据文件生成目录（根据端口名修改）  dir /app/aimsg/redis\_cluster/7000/data/ |

PS：其它机器以及其它目录7000 7010，redis\_70\*0.conf内容修改同上原理。

1. 启动redis节点

|  |
| --- |
| 将三台主机的redis节点启动  ~/redis/bin/redis-server ~/redis\_cluster/7000/redis\_7000.conf  ~/redis/bin/redis-server ~/redis\_cluster/7010/redis\_7010.conf  ~/redis/bin/redis-server ~/redis\_cluster/7100/redis\_7100.conf  ~/redis/bin/redis-server ~/redis\_cluster/7110/redis\_7110.conf  ~/redis/bin/redis-server ~/redis\_cluster/7200/redis\_7200.conf  ~/redis/bin/redis-server ~/redis\_cluster/7210/redis\_7210.conf  检查日志是否已经启动，日志目录~/redis\_cluster/端口/data/redis-log-xxxx.log  tail -200f ~/redis\_cluster/7000/data/redis-log-7000.log |

1. 查看redis进程

|  |
| --- |
| ps -ef | grep redis |

注意：这时候启动的只是单机启动，还没有关联到一起，要把各个单机的redis组成集群模式还需要ruby工具来帮忙。

## 安装Ruby

弃坑：ruby 2.4.3安装在阿里云esc机器有bug，运行后出现该错误

|  |
| --- |
| [ban@ban ruby]$ ./ruby  <internal:gem\_prelude>:4:in `require': cannot load such file -- rubygems.rb (LoadError)  from <internal:gem\_prelude>:4:in `<internal:gem\_prelude>' |

最新安装包：ruby-2.6.3.tar.gz

* 1. 拷贝安装包ruby-2.6.3.tar.gzz到soft目录，进行解压

|  |
| --- |
| tar -zxvf ./ruby-2.6.3.tar.gz  mv ./ruby-2.6.3 ~/ |

* 1. 编译安装到指定的目录，否则默认安装到root用户下

|  |
| --- |
| cd /home/ban/ruby  #开始指定目录编译（注意：prefix有时候起不来作用，有时可以，虽然每次指定另外路径，结果还是以当前运行的目录解压，所以最好指定当前ruby的解压目录和别的目录，反正指定后还是当前目录）  ./configure --prefix=/home/ban/ruby --enable-pthread  make  make install |

* 1. 执行configure 出错的话（不能切换目录错误提示），需要切换目录再执行语句

|  |
| --- |
| 执行：./configure --prefix=/home/ban/ruby --enable-pthread  #出现报错    #说明不能指定当前目录  #注意：ruby-2.6.3有个bug，--prefix指定了目录后，有时候出现安装在运行命令的当前目录。比如说当前命令在/home/ban/ruby 执行，但是--prefix=/home/ban/rubybak，/home/ban/ruby执行make install后还会编译到了/home/ban/ruby。所以这里修改为/home/ban/rubybak。  #回到根目录进行执行  [ban@ban ruby-2.6.3]$ cd /home/ban/ruby-2.6.3/  ~/configure --prefix=/home/ban/rubybak --enable-pthread  #编译完成 |

## redis-4.1.2.gem安装

安装安装集权启动脚本依赖redis-4.1.2.gem，类似于java中的jar

* 1. 必须拷贝redis-4.1.2.gem 到/home/ban/ruby-2.6.3/bin目录，执行gem安装

|  |
| --- |
| cd /home/ban/ruby-2.6.3/bin;  ./gem install redis-4.1.2.gem --user-instal |

* 1. 报错解决

|  |
| --- |
| 问题执行报错了    提示路径格式有问题  解决：  修改/home/ban/ruby/bin/gem脚本  开头修改为具体绝对路径，必须绝对路径，相对路径执行报错，并且查找不到  #!/usr/bin/env /home/ban/ruby-2.6.3/ruby |

* 1. 重新执行gem，安装 redis-4.1.2.gem

|  |
| --- |
| cd /home/ban/ruby-2.6.3/bin;  ./gem install --local ~/soft/redis/redis-4.1.2.gem  但是报错了    [ban@ban bin]$ ./gem install ./redis-4.1.2.gem --user-install  ERROR: Loading command: install (LoadError)  cannot load such file -- zlib  ERROR: While executing gem ... (NoMethodError)  undefined method `invoke\_with\_build\_args' for nil:NilClass |

* 1. 解决gem安装报错

安装zlib-1.2.11.tar.gz安装包

|  |
| --- |
| tar zxvf ./zlib-1.2.11.tar.gz  mv ./zlib-1.2.11 ~/zlib  cd /home/ban/zlib/  ./configure --prefix=/home/ban/zlib/  make && make install |

给ruby增加zlib依赖

|  |
| --- |
| 1、添加依赖并安装  cd /home/ban/ruby-2.6.3/ext/zlib  ~/ruby-2.6.3/ruby ./extconf.rb --with-zlib-dir=/home/ban/zlib/  make  make install  执行make命令报错    [ban@ban zlib]$ make  make: \*\*\* No rule to make target `/include/ruby.h', needed by `zlib.o'. Stop.  （找不到ruby.h，只能手动修改为绝对路径）   1. 修改/home/ban/ruby-2.6.3/ext/zlib/Makefile   修改前    修改该包的绝对路径: /home/ban/ruby-2.6.3/include/ruby.h     1. 重新编译   make && make install |

* 1. 重新执行gem，安装 redis-4.1.2.gem

|  |
| --- |
| cd /home/ban/ruby-2.6.3/bin;  ./gem install --local ~/soft/redis/redis-4.1.2.gem    安装成功 |

## 配置集群

1. 集群主备规划

|  |
| --- |
| 172.16.239.107 部署：master-172.16.239.107:7000 , slave-172.16.239.107:7010  172.16.239.107 部署：master-172.16.239.107:7100 , slave-172.16.239.107:7110  172.16.239.107 部署：master-172.16.239.107:7200 , slave-172.16.239.107:7210 |

1. 构建集群（伪集群，因为现在只有一台机器）

|  |  |
| --- | --- |
| ~/ruby-2.6.3/ruby ~/redis/src/redis-trib.rb create --replicas 1 172.16.239.107:7000 172.16.239.107:7010 172.16.239.107:7100 172.16.239.107:7110 172.16.239.107:7200 172.16.239.107:7210  1、该redis必须先启动，否则会报连接不上的错误     1. 启动redis各个节点      1. 启动成功      |  | | --- | | [ban@ban data]$ ~/ruby-2.6.3/ruby ~/redis/src/redis-trib.rb create --replicas 1 172.16.239.107:7000 172.16.239.107:7010 172.16.239.107:7100 172.16.239.107:7110 172.16.239.107:7200 172.16.239.107:7210  ##执行结果：  /home/ban/redis/src/redis-trib.rb:1573: warning: key "threshold" is duplicated and overwritten on line 1573  >>> Creating cluster  >>> Performing hash slots allocation on 6 nodes...  Using 3 masters:  172.16.239.107:7000  172.16.239.107:7010  172.16.239.107:7100  Adding replica 172.16.239.107:7110 to 172.16.239.107:7000  Adding replica 172.16.239.107:7200 to 172.16.239.107:7010  Adding replica 172.16.239.107:7210 to 172.16.239.107:7100  M: c43333d0e74925b1953e9b1b46f97aefcd4c8d8e 172.16.239.107:7000  slots:0-5460 (5461 slots) master  M: 8a2b1a835644969129e86a321c3178c1aae4b810 172.16.239.107:7010  slots:5461-10922 (5462 slots) master  M: 4062245bef0c2d81acc76225671a1cd8b7229d62 172.16.239.107:7100  slots:10923-16383 (5461 slots) master  S: 39ffd6419f139721c996afa3ad6641da26bc6ba2 172.16.239.107:7110  replicates c43333d0e74925b1953e9b1b46f97aefcd4c8d8e  S: eaeaf4afcb9bec71468df0f2e1f02f02031d5788 172.16.239.107:7200  replicates 8a2b1a835644969129e86a321c3178c1aae4b810  S: 771e567c88335be5db9c65c05409891ddd7b69a5 172.16.239.107:7210  replicates 4062245bef0c2d81acc76225671a1cd8b7229d62  Can I set the above configuration? (type 'yes' to accept): yes  >>> Nodes configuration updated  >>> Assign a different config epoch to each node  >>> Sending CLUSTER MEET messages to join the cluster  Waiting for the cluster to join....  >>> Performing Cluster Check (using node 172.16.239.107:7000)  M: c43333d0e74925b1953e9b1b46f97aefcd4c8d8e 172.16.239.107:7000  slots:0-5460 (5461 slots) master  M: 8a2b1a835644969129e86a321c3178c1aae4b810 172.16.239.107:7010  slots:5461-10922 (5462 slots) master  M: 4062245bef0c2d81acc76225671a1cd8b7229d62 172.16.239.107:7100  slots:10923-16383 (5461 slots) master  M: 39ffd6419f139721c996afa3ad6641da26bc6ba2 172.16.239.107:7110  slots: (0 slots) master  replicates c43333d0e74925b1953e9b1b46f97aefcd4c8d8e  M: eaeaf4afcb9bec71468df0f2e1f02f02031d5788 172.16.239.107:7200  slots: (0 slots) master  replicates 8a2b1a835644969129e86a321c3178c1aae4b810  M: 771e567c88335be5db9c65c05409891ddd7b69a5 172.16.239.107:7210  slots: (0 slots) master  replicates 4062245bef0c2d81acc76225671a1cd8b7229d62  [OK] All nodes agree about slots configuration.  >>> Check for open slots...  >>> Check slots coverage...  [OK] All 16384 slots covered. |   ps：--replicas 1的意思是每个master有1个slave。 |

注意1：要用~/ruby-2.6.3/ruby 不要用~/ruby-2.6.3/bin/ruby

否则会报错

|  |
| --- |
| [ban@ban bin]$ ~/ruby-2.6.3/bin/ruby ~/redis/src/redis-trib.rb create --replicas  /home/ban/redis/src/redis-trib.rb:1573: warning: key "threshold" is duplicated and overwritten on line 1573  Traceback (most recent call last):  2: from /home/ban/redis/src/redis-trib.rb:25:in `<main>'  1: from /usr/local/lib/ruby/2.6.0/rubygems/core\_ext/kernel\_require.rb:54:in `require'  /usr/local/lib/ruby/2.6.0/rubygems/core\_ext/kernel\_require.rb:54:in `require': cannot load such file -- redis (LoadError) |

~~注意2：搭建集群至少需要3主3从6台服务器或进程，否则你会遇到这错误~~

|  |
| --- |
| ~~[ban@ban ~]$ ~/ruby-2.6.3/bin/ruby ~/redis/src/redis-trib.rb~~  ~~/home/ban/redis/src/redis-trib.rb:1573: warning: key "threshold" is duplicated and overwritten on line 1573~~  ~~Traceback (most recent call last):~~  ~~2: from /home/ban/redis/src/redis-trib.rb:25:in `<main>'~~  ~~1: from /usr/local/lib/ruby/2.6.0/rubygems/core\_ext/kernel\_require.rb:54:in `require'~~  ~~/usr/local/lib/ruby/2.6.0/rubygems/core\_ext/kernel\_require.rb:54:in `require': cannot load such file -- redis (LoadError)~~ |

注意3：ip一定要写对，否则会出现一直等待的情况

|  |
| --- |
| ip 172开头达成72开头，又没有报错信息，等了好久 |

1. 查看集群信息

|  |
| --- |
| 1. 登录其中一个redis node   cd /home/ban/redis/bin;  ./redis-cli -h 172.16.239.107 -c -p 7100  参数：-h：配置文件中指定的ip地址  -c：以集群方式登录  -p：指定端口登录    注意：这里的ip必须是启动集群时指定的ip（由于配置文件指定了ip，必须以指定ip登录），否则连接失败。   1. 查看集群信息   172.16.239.107:7100> cluster info     1. 查看集群的主从信息（关联关系）   172.16.239.107:7100> cluster nodes |

1. 手动分配集群的主从关系

|  |
| --- |
| 举例，因为当前只是在一台机器，没有该情况：由于随机分配导致30主机的主节点从节点都是自己，我们需要手动调整。  1、出现问题的场景：（30机器指定主从节点都是在一台机器，如果30机器宕机，就会存在本来有3主节点可以存活三个主节点（从升级为主），现在30机器宕机的话，就只能两个主节点，达不到高可用，导致部分数据查不到的情况，因为数据是分不同节点保存的。）     1. 解决：手动调整关系   a.登录方式指定要更改隶属关系的IP、端口  ./redis-cli -c -h 172.20.35.28 -p 7010  b.更改隶属关系时，直接指定新的master节点的ID  执行命令：cluster replicate b7dab3028c8ddb5a92cb3aeb454076d403d7d38f  c.执行效果（见下图）：这样当30机器宕机后，其他机器的从节点可升级为主节点，其他机器的主节点就照样还是主节点，保持了三个node的存在。     1. 具体可以参考营销中心部署文档 2. 主从分配原则（前3为master，后3为slave）   观察发现多个部署文档发现，redis-trib.rb分配主从关系并不是随机的  示例1、      示例2：    根据上图来说，6个节点的情况下，前3个端口为master节点，后3个端口为slave节点。  如果是12个节点，1-3节点为master节点，4-6为salve节点，7-9节点为master节点，10-12为salve节点。 |

1. 测试是否达到集群效果（检查数据是否可以在不同node查找到数据）

|  |
| --- |
| 1. 在172.16.239.107:7100节点保存数据   172.16.239.107:7100> set foo bar  OK  172.16.239.107:7100> get foo  "bar"  172.16.239.107:7100>     1. 登录172.16.239.107:7200检查是否可以追踪到7100node保存的数据   [ban@ban bin]$ ./redis-cli -h 172.16.239.107 -c -p 7200  172.16.239.107:7200> ping  PONG  172.16.239.107:7200> get foo  -> Redirected to slot [12182] located at 172.16.239.107:7100  "bar"  172.16.239.107:7100> get foo  "bar"  172.16.239.107:7100> |

1. 设置集群密码（永久方式）

|  |  |
| --- | --- |
| 1. 修改所有Redis集群中的redis.conf文件加入：    masterauth passwd123 (用于让slave能连上master，不配置的话会出现同步数据失败)  requirepass passwd123  说明：这种方式需要重新启动各节点  注意：masterauth必须配置，否则会报这个错，网络找到的示例   |  | | --- | | 4939:S 25 Jan 09:53:20.450 # MASTER aborted replication with an error: NOAUTH Authentication required.  4939:S 25 Jan 09:53:21.291 \* Connecting to MASTER 120.27.137.142:6379  4939:S 25 Jan 09:53:21.292 \* MASTER <-> SLAVE sync started  4939:S 25 Jan 09:53:21.317 \* Non blocking connect for SYNC fired the event.  4939:S 25 Jan 09:53:21.342 \* Master replied to PING, replication can continue...  4939:S 25 Jan 09:53:21.368 \* (Non critical) Master does not understand REPLCONF listening-port: -NOAUTH Authentication required.  4939:S 25 Jan 09:53:21.393 \* (Non critical) Master does not understand REPLCONF capa: -NOAUTH Authentication required.  4939:S 25 Jan 09:53:21.393 \* Partial resynchronization not possible (no cached master)  4939:S 25 Jan 09:53:21.419 # Unexpected reply to PSYNC from master: -NOAUTH Authentication required.  4939:S 25 Jan 09:53:21.419 \* Retrying with SYNC...  4939:S 25 Jan 09:53:21.444 # MASTER aborted replication with an error: NOAUTH Authentication required. |   2、修改示例  cd /home/ban/redis\_cluster/7000;  vim ./redis\_7000.conf ;  修改权限配置  masterauth 123456  requirepass 123456     1. （重复修改第1、2步的node，直到修改了所有node，这里不做示例）然后重启redis所有修改的node   ~/redis/bin/redis-server ~/redis\_cluster/7000/redis\_7000.conf   1. 重新登录（不使用密码的情况下）   [ban@ban bin]$ ./redis-cli -h 172.16.239.107 -c -p 7000  172.16.239.107:7000> pin g  (error) ERR unknown command 'pin'  172.16.239.107:7000> ping  (error) NOAUTH Authentication required.  ##登录失败     1. 重新登录（使用密码登录的情况下）   [ban@ban bin]$ ./redis-cli -h 172.16.239.107 -c -p 7000 -a 123456  参数：-a：配置的密码进行登录  172.16.239.107:7000> ping  PONG  172.16.239.107:7000> |

1. 关于redis配置密码的疑问

|  |
| --- |
| 其次，关于启用redis密码认证的涉及的几个问题：  1、是否只设置requirepass就可以？masterauth是否需要同步设置？  答案：redis启用密码认证一定要requirepass和masterauth同时设置。  如果主节点设置了requirepass登录验证，在主从切换，slave在和master做数据同步的时候首先需要发送一个ping的消息给主节点判断主节点是否存活，再监听主节点的端口是否联通，发送数据同步等都会用到master的登录密码，否则无法登录，log会出现响应的报错。也就是说slave的masterauth和master的requirepass是对应的，所以建议redis启用密码时将各个节点的masterauth和requirepass设置为相同的密码，降低运维成本。当然设置为不同也是可以的，注意slave节点masterauth和master节点requirepass的对应关系就行。  2、requreipass和master的作用？  masterauth作用：主要是针对master对应的slave节点设置的，在slave节点数据同步的时候用到。  requirepass作用：对登录权限做限制，redis每个节点的requirepass可以是独立、不同的。 |

# 七、zookeeper、kafka等中间件启停脚本

