# 实现数据库高可用

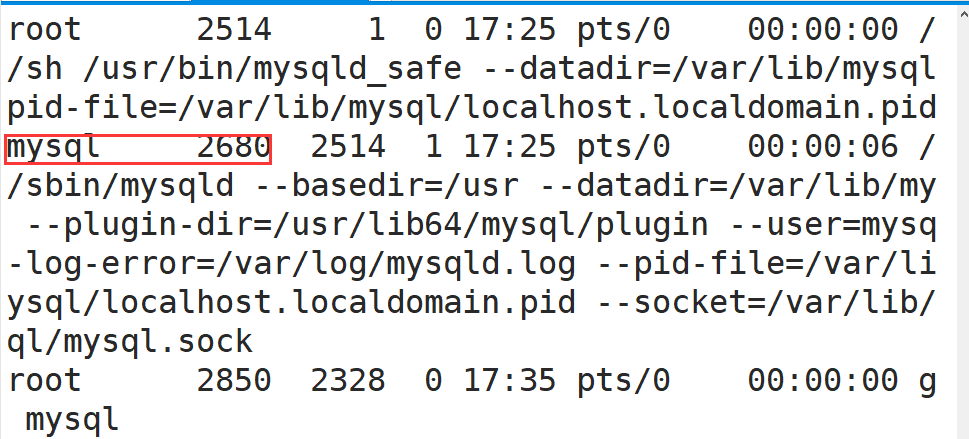
## 配置数据库主从

### 关于数据库启动报错

1. PID报错/socket报错

ps -ef|greap mysql

2.查找Mysql后的PID号



3.之后杀死进程

Kill -9 2680

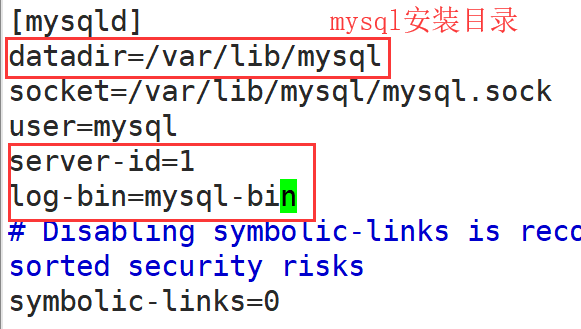
4.重启mysql服务器

service mysql start即可

## 主从配置实现步骤

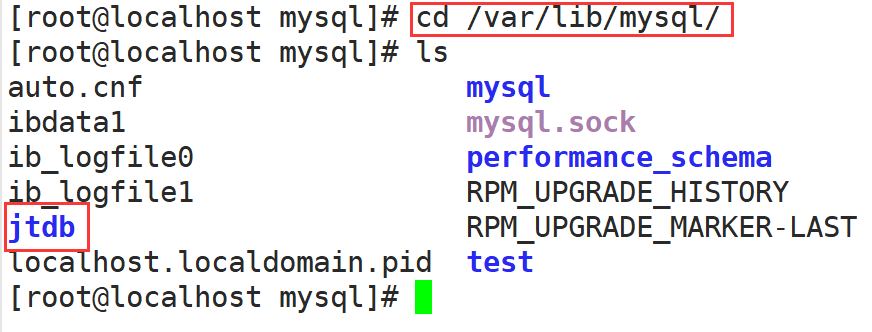
### 修改主库mysql配置文件my.cnf

命令: vim /etc/my.cnf

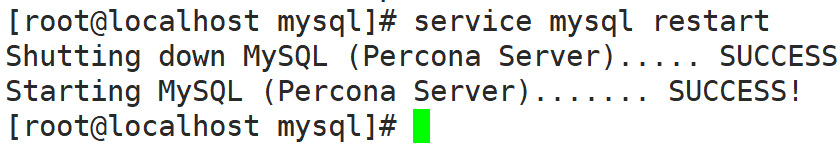


### Mysql文件结构

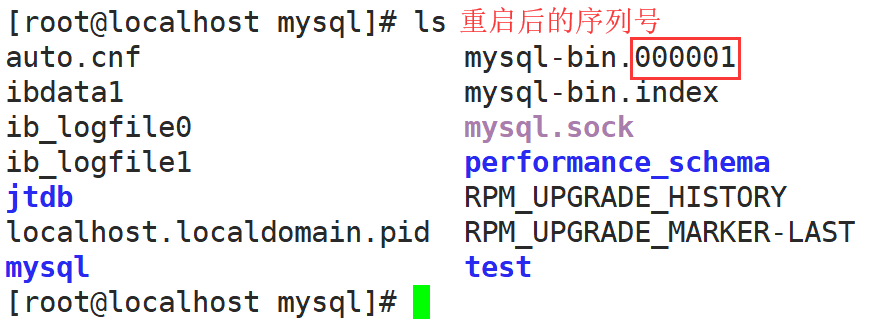
1. 配置二进制之前的文件



1. 重启mysql服务



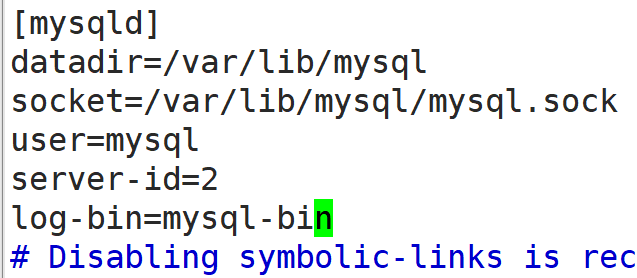
1. 重启后mysql数据库



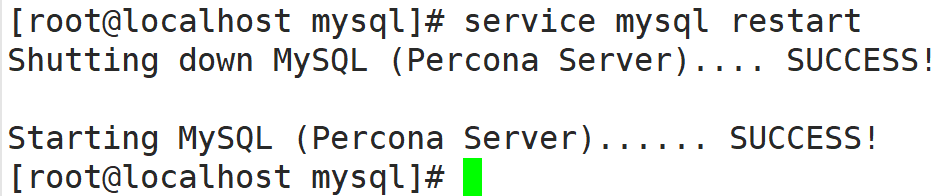
### 修改从库的my.cnf

编辑从库配置文件

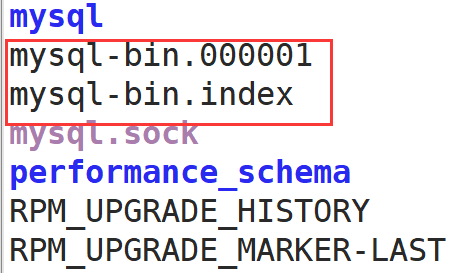
vim /etc/my.cnf



2.重启mysql数据库

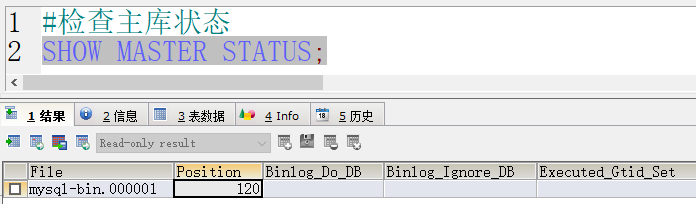


3.检测二进制日志文件是否生效



### 主从挂载命令

1. 检查主库状态信息



1. 实现主从挂载

#从库实现主从挂载

SHOW MASTER STATUS;

#IP地址/端口号/用户名/密码/配置主库二进制文件/位置

CHANGE MASTER TO MASTER\_HOST="192.168.126.162",

MASTER\_PORT=3306,MASTER\_USER="root",

MASTER\_PASSWORD="root",

MASTER\_LOG\_FILE="mysql-bin.000001",

MASTER\_LOG\_POS=120

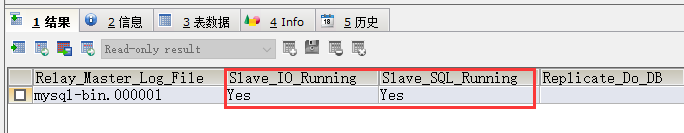
#启动主从服务

START SLAVE;

#检查主从服务状态

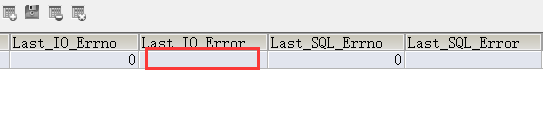
SHOW SLAVE STATUS;

1. 检测状态信息



### 报错

1. 检查日志



1. 检查my.cnf文件是否正确
2. 检查server-id是否不一致(对)
3. 检查数据库是否重启
4. 检查防火墙是否关闭.

重新搭建主从的步骤:

#第一步先关闭主从服务

STOP SLAVE

#.实现主从挂载

CHANGE MASTER TO MASTER\_HOST="192.168.126.162",

MASTER\_PORT=3306,MASTER\_USER="root",

MASTER\_PASSWORD="root",

MASTER\_LOG\_FILE="mysql-bin.000001",

MASTER\_LOG\_POS=120

#启动主从服务

START SLAVE

#检查主从状态

SHOW SLAVE STATUS

## 实现数据库读写分离

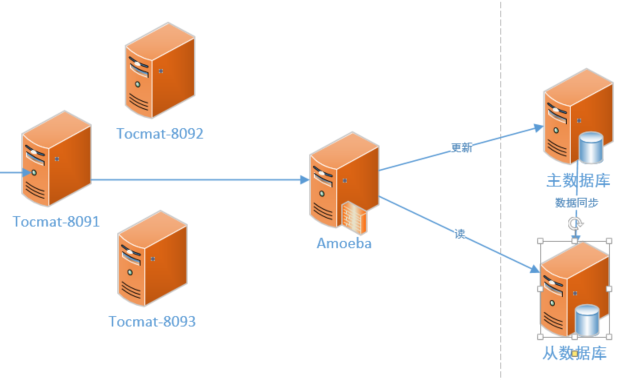
### 业务需求

如果在服务中,只配置主库,如果遇到了高并发的现象,则主库有宕机的风险.用户大量的请求中,查询的数量是最多的.如何优化.

### 使用代理服务器

Amoeba介绍:

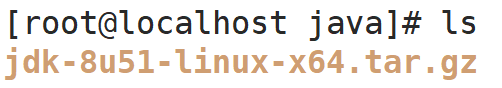
Amoeba是一个以MySQL为底层数据存储，并对应用提供MySQL协议接口的proxy。它集中地响应应用的请求，依据用户事先设置的规则，将SQL请求发送到特定的数据库上执行。基于此可以实现**负载均衡**、[**读写分离**](https://baike.baidu.com/item/%E8%AF%BB%E5%86%99%E5%88%86%E7%A6%BB)、**高可用性**等需求。与MySQL官方的MySQL Proxy相比，作者强调的是amoeba配置的方便（基于XML的配置文件，用SQLJEP语法书写规则，比基于lua脚本的MySQL Proxy简单）。



## 搭建Amoeba步骤

### 配置JDK

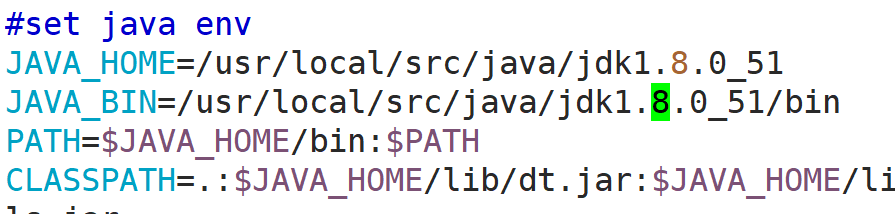
1. 上传安装文件



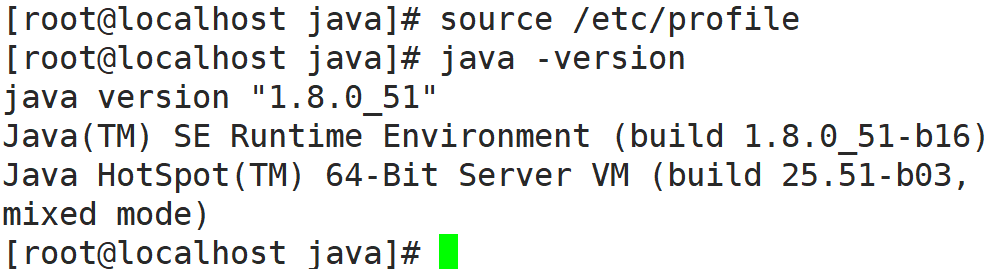
1. 解压文件

tar -xvf jdk-8u51-linux-x64.tar.gz

1. 修改环境变量

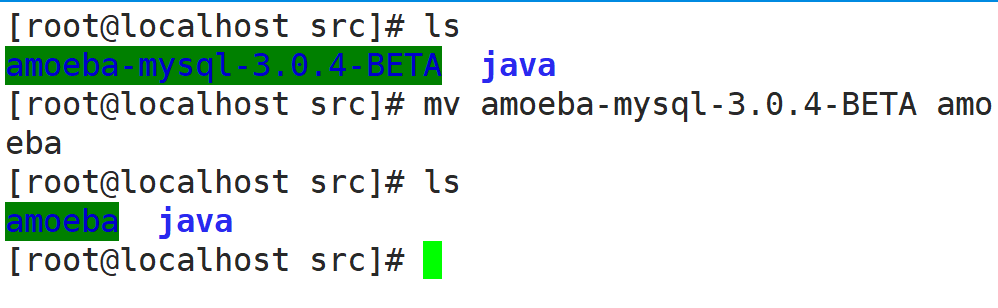


1. 重新加载环境变量

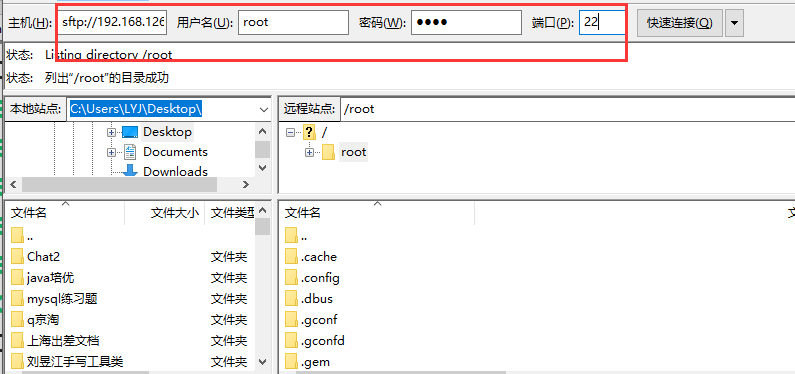


### 安装Amoeba

1. 解压文件

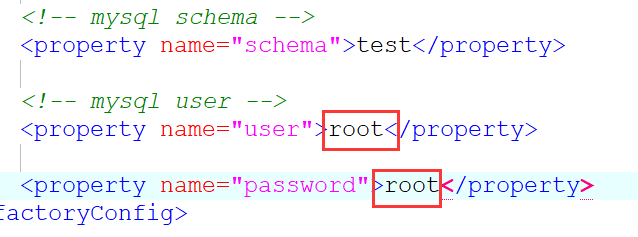


### 配置FZ工具



### 修改dbServices.xml

1.配置公有的数据库用户名和密码



2.配置负载均衡

<!--配置数据库-->

<dbServer name="master" parent="abstractServer">

<factoryConfig>

<!-- mysql ip -->

<property name="ipAddress">192.168.126.162</property>

</factoryConfig>

</dbServer>

<dbServer name="slave01" parent="abstractServer">

<factoryConfig>

<!-- mysql ip -->

<property name="ipAddress">192.168.126.163</property>

</factoryConfig>

</dbServer>

<!--实现负载均衡 主读-->

<dbServer name="multiPool" virtual="true">

<poolConfig class="com.meidusa.amoeba.server.MultipleServerPool">

<!-- Load balancing strategy: 1=ROUNDROBIN , 2=WEIGHTBASED 挑选不忙的服务器访问 , 3=HA-->

<property name="loadbalance">1</property>

<!-- Separated by commas,such as: server1,server2,server1 -->

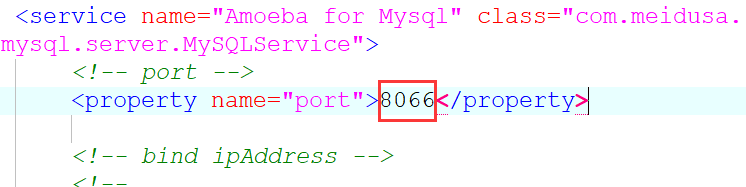
<property name="poolNames">slave01,master,slave01</property>

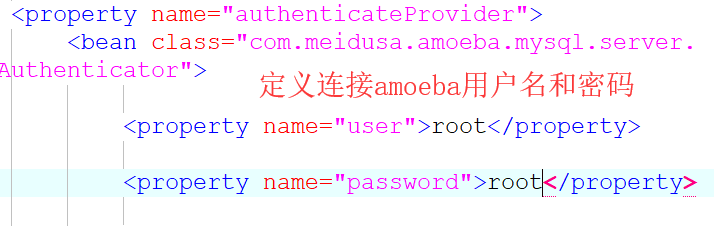
</poolConfig>

</dbServer>

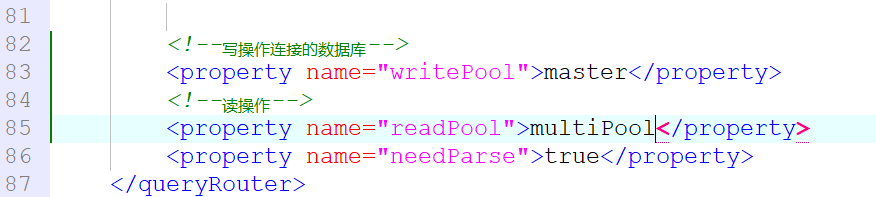
### 编辑Amoeba.xml

1. 修改端口号



2. 

3.配置读写分离策略



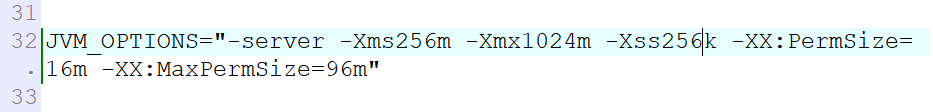
### 修改JVM内存

JVM\_OPTIONS="-server -Xms256m -Xmx1024m -Xss256k -XX:PermSize=16m -XX:MaxPermSize=96m"

-Xms256m :jvm初始化内存大小

-Xmx1024m :JVM最大内存空间

-Xss196k :单个线程的大小

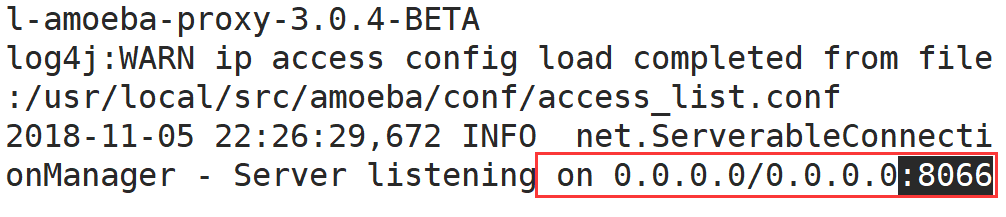


### 读写分离测试

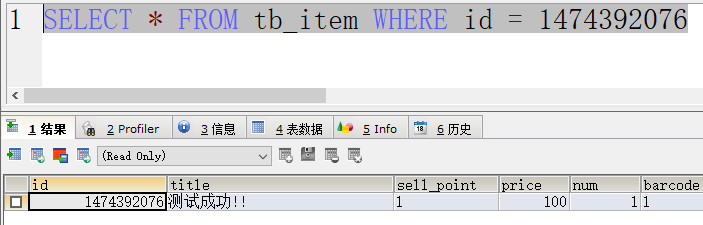
1. 启动amoeba

./launcher

./shutdown



1. 修改从库信息,检查主从同步负载均衡是否生效

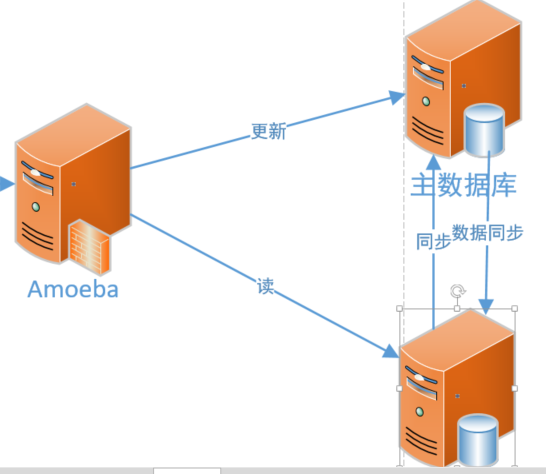


## 双机热备

### 说明:

Master:A 主要的作用向从库B同步数据.

从库B:向A主库同步数据.



调用流程:

1. 正常情况下,主库负责写库操作.从库主要负责读的操作.
2. 当主库宕机时,通过某些技术手段可以实现数据库的主从切换.这时从库即充当主库又充当从库.
3. 运维人员将主库修复完成后启动.因为之前已经配置了双机热备的形式.这时因为从库已经运行了一段时间,其中的数据一定比主库的多,这时从库将数据同步给主库,最终实现了数据的一致.

### 实现挂载

注意事项:挂载的数据库中的数据一定要保持一致.

#实现从库反向挂载

CHANGE MASTER TO MASTER\_HOST="192.168.126.163",

MASTER\_PORT=3306,MASTER\_USER="root",

MASTER\_PASSWORD="root",

MASTER\_LOG\_FILE="mysql-bin.000001",

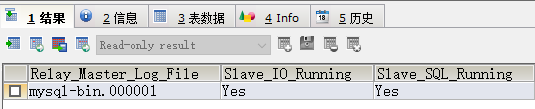
MASTER\_LOG\_POS=381;

#启动主从服务

START SLAVE

#检查状态

SHOW SLAVE STATUS

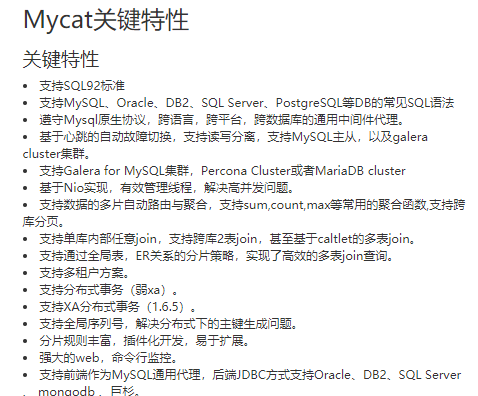


## MyCat数据库代理

### Mycat介绍



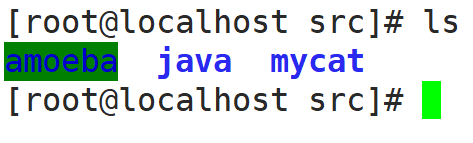
### 特性



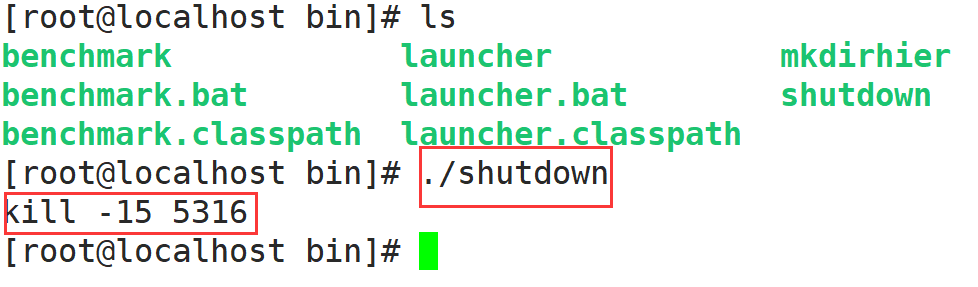
## Mycat安装

### 解压安装包

tar -xvf Mycat-server-1.7.0-DEV-20170416134921-linux.tar.gz

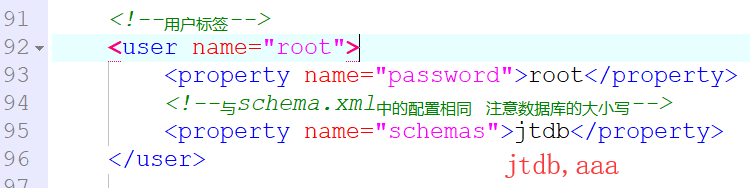


注意事项:关闭amoeba服务器



### 修改server.xml

说明:该文件主要定义用户.同时定义当前用户可以访问哪些数据库.



### 修改schema.xml

说明:该配置文件主要的作用.定义用户和数据表之间的关联关系(逻辑库/逻辑表)

<?xml version="1.0"?>

<!DOCTYPE mycat:schema SYSTEM "schema.dtd">

<mycat:schema xmlns:mycat="http://io.mycat/">

<!--name属性是自定义的 dataNode表示数据库的节点信息-->

<schema name="jtdb" checkSQLschema="false" sqlMaxLimit="100" dataNode="jtdb"/>

<!--定义节点名称/节点主机/数据名称-->

<dataNode name="jtdb" dataHost="localhost1" database="jtdb" />

<!--参数介绍-->

<!--balance 0表示所有的读操作都会发往writeHost主机 -->

<!--1表示所有的读操作发往readHost和闲置的主节点中-->

<!--writeType=0 所有的写操作都发往第一个writeHost主机-->

<!--writeType=1 所有的写操作随机发往writeHost中-->

<!--dbType 表示数据库类型 mysql/oracle-->

<!--dbDriver="native" 固定参数 不变-->

<!--switchType=-1 表示不自动切换, 主机宕机后不会自动切换从节点-->

<!--switchType=1 表示会自动切换(默认值)如果第一个主节点宕机后,Mycat会进行3次心跳检测,如果3次都没有响应,则会自动切换到第二个主节点-->

<!--并且会更新/conf/dnindex.properties文件的主节点信息 localhost1=0 表示第一个节点.该文件不要随意修改否则会出现大问题-->

<dataHost name="localhost1" maxCon="1000" minCon="10" balance="1"

writeType="0" dbType="mysql" dbDriver="native" switchType="1" slaveThreshold="100">

<heartbeat>select 1</heartbeat>

<!--配置第一台主机主要进行写库操作,在默认的条件下Mycat主要操作第一台主机在第一台主机中已经实现了读写分离.因为默认写操作会发往137的数据库.读的操作默认发往141.如果从节点比较忙,则主节点分担部分压力.

-->

<writeHost host="hostM1" url="192.168.126.162:3306" user="root" password="root">

<!--读数据库-->

<readHost host="hostS1" url="192.168.126.163:3306" user="root" password="root" />

</writeHost>

<!--定义第二台主机 由于数据库内部已经实现了双机热备.-->

<!--Mycat实现高可用.当第一个主机137宕机后.mycat会自动发出心跳检测.检测3次.-->

<!--如果主机137没有给Mycat响应则判断主机死亡.则回启东第二台主机继续为用户提供服务.-->

<!--如果137主机恢复之后则处于等待状态.如果141宕机则137再次持续为用户提供服务.-->

<!--前提:实现双机热备.-->

<writeHost host="hostM2" url="192.168.126.163:3306" user="root" password="root">

<readHost host="hostS1" url="192.168.126.162:3306" user="root" password="root" />

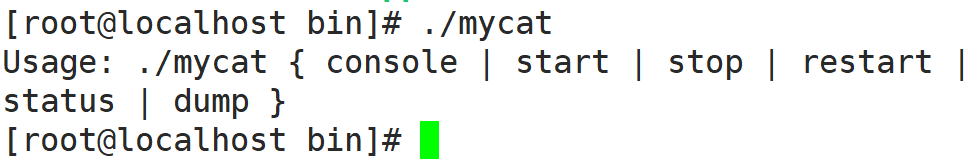
</writeHost>

</dataHost>

</mycat:schema>

### 实现Mycat数据库高可用

1. 将课前资料中的文件修改好之后上传到服务器中.
2. 启动mycat



Console:控制台信息

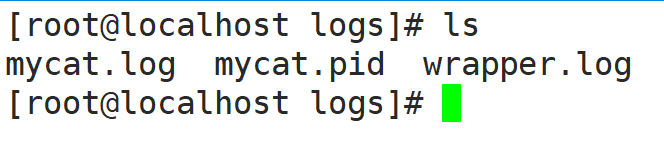
Start:启动

Stop:停止

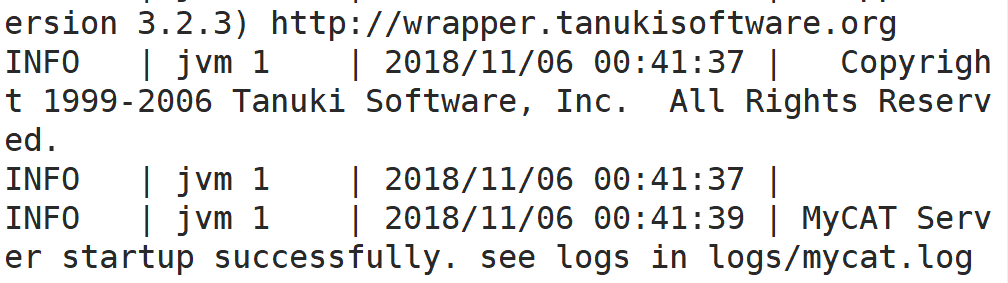
Restart:重启

Status: 状态信息.

### 检查日志



检查日志:



### 数据库高可用检测

1. 将数据库主库宕机.检查用户访问是否正确.
2. 用户更新数据,检查是否正常.
3. 将主库重启.检查数据是否同步.

