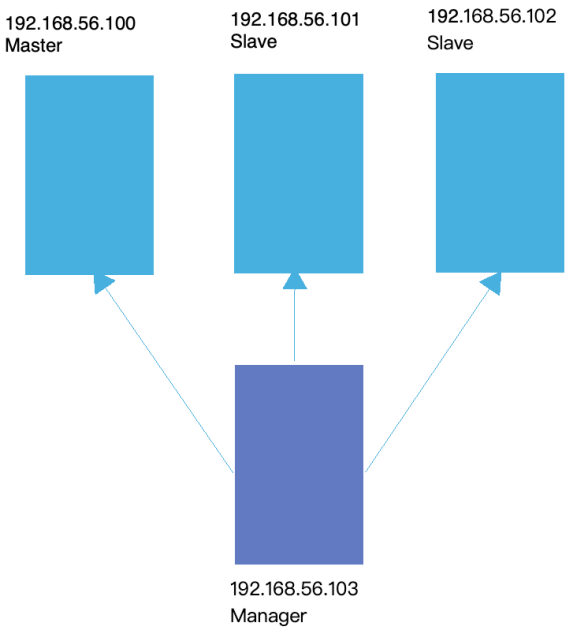
1. 环境软件版本

环境&软件	版本
虚拟机&VirtualBox	6.1
服务器&CentOS	7.8
数据库&MySQL	5.7.31
远程连接&ZenTermLite	4.1.0
远程文件传输&SCP	

2. 环境架构介绍

架构如图所示，4台机器的IP和角色如下表：

机器名称	IP	角色	权限
master100	192.168.56.100	数据库Master	可读写、主库
slave101	192.168.56.101	数据库Slave	只读、从库
slave102	192.168.56.102	数据库Slave	只读、从库
mha103	192.168.56.103	MHA Manager	高可用监控



3. MySQL主从搭建

3.1 MySQL下载安装（3台：1主2从）

下载mysql数据库包

```
wget https://dev.mysql.com/get/Downloads/MySQL-5.7/mysql-5.7.31-
1.el7.x86_64.rpm-bundle.tar
```

解压数据包

```
tar -xvf mysql-5.7.31-1.el7.x86_64.rpm-bundle.tar
```

安装mysql

CentOS 7自带mariadb，查看

```
rpm -qa|grep mariadb
```

移除mariadb

```
rpm -e mariadb-libs-5.5.65-1.el7.x86_64 --nodeps
```

按顺序依次安装mysql依赖包

```
rpm -ivh mysql-community-common-5.7.31-1.el7.x86_64.rpm
rpm -ivh mysql-community-libs-5.7.31-1.el7.x86_64.rpm
rpm -ivh mysql-community-libs-compat-5.7.31-1.el7.x86_64.rpm
rpm -ivh mysql-community-client-5.7.31-1.el7.x86_64.rpm
```

> [root@mha103 ~]# rpm -ivh mysql-community-server-5.7.31-1.el7.x86_64.rpm
> 警告：mysql-community-server-5.7.31-1.el7.x86_64.rpm: 头V3 DSA/SHA1 Signature, 密钥 ID 5072e1f5: NOKEY
> 错误：依赖检测失败：
> /usr/bin/perl 被 mysql-community-server-5.7.31-1.el7.x86_64 需要
> net-tools 被 mysql-community-server-5.7.31-1.el7.x86_64 需要
> perl(Getopt::Long) 被 mysql-community-server-5.7.31-1.el7.x86_64 需要
> perl(strict) 被 mysql-community-server-5.7.31-1.el7.x86_64 需要
>
> 出现报错，需要先为mysql-community-server-5.7.31-1.el7.x86_64安装perl和net-tools

```
yum install perl
```

```
yum install net-tools
```

```
rpm -ivh mysql-community-server-5.7.31-1.el7.x86_64.rpm
rpm -ivh mysql-community-devel-5.7.31-1.el7.x86_64.rpm
```

初始化mysql

```
mysqld --initialize --user=mysql
```

从日志中获取随机生成的用户临时密码: Y.E#Noi-/4kJ

```
cat /var/log/mysqld.log
```

> 2020-08-23T00:44:34.269924Z 0 [Warning] TIMESTAMP with implicit DEFAULT value is deprecated. Please use --explicit_defaults_for_timestamp server option (see documentation for more details).
> 2020-08-23T00:44:34.420157Z 0 [Warning] InnoDB: New log files created, LSN=45790
> 2020-08-23T00:44:34.452736Z 0 [Warning] InnoDB: Creating foreign key constraint system tables.
> 2020-08-23T00:44:34.515167Z 0 [Warning] No existing UUID has been found, so we assume that this is the first time that this server has been started. Generating a new UUID: d07d9fc6-e4d9-11ea-b896-08002713cbf1.
> 2020-08-23T00:44:34.518219Z 0 [Warning] Gtid table is not ready to be used. Table 'mysql.gtid_executed' cannot be opened.
> 2020-08-23T00:44:35.539023Z 0 [Warning] CA certificate ca.pem is self signed.
> 2020-08-23T00:44:35.625459Z 1 [Note] A temporary password is generated for

> root@localhost: **Y.E#Noi-/4kJ**

启动mysql，运行在后台，之后连接虚拟机会自动启动

```
systemctl start mysqld.service
```

查看状态

```
systemctl status mysqld.service
```

登录mysql

```
mysql -uroot -p
Enter password: Y.E#Noi-/4kJ
```

进入mysql，重置密码为root

```
mysql> set password=password('root');
```

之后就可以用root为密码登录了

```
mysql> exit;
```

```
[root@mha103 ~]# mysql -uroot -p
Enter password: root
```

> 如果遇到：
>
> mysql> SET PASSWORD = PASSWORD('root');
> ERROR 1819 (HY000): Your password does not satisfy the current policy requirements
>
> ```
> mysql> SET GLOBAL validate_password_policy = 0;
> mysql> uninstall plugin validate_password;
> ```

3.2 关闭防火墙

不同的MySQL直接要互相访问，需要关闭linux的防火墙，否则就要在配置/etc/sysconfig/iptables中增加规则。配置防火墙不是本次作业的重点，所以四台服务器均关闭防火墙。

```
systemctl stop firewalld
```

```
systemctl disable firewalld.service
```

3.3 MySQL主从配置

Master节点

修改Master配置文件

```
vi /etc/my.cnf
```

```
#bin_log配置
log_bin=mysql-bin
server-id=1
sync-binlog=1
binlog-ignore-db=information_schema
binlog-ignore-db=mysql
binlog-ignore-db=performance_schema
binlog-ignore-db=sys
#relay-log配置
relay_log=mysql-relay-bin
log_slave_updates=1
relay_log_purge=0
#gtid配置
#开启gtid
#gtid_mode=on
#enforce_gtid_consistency=1
```

重启服务

```
systemctl restart mysqld
```

主库给从库授权

在MySQL命令行执行如下命令：

```
grant replication slave on *.* to root@'%' identified by 'root';
grant all privileges on *.* to root@'%' identified by 'root';

flush privileges;
show master status; //查看主库master_log_file='mysql-bin.000006',
master_log_pos=154
```

Slave节点

修改Slave配置文件，两台Slave的server-id分别设置为2和3

```
vi /etc/my.cnf
```

```
#bin_log配置
log_bin=mysql-bin
#服务器ID，从库1是2，从库2是3
server-id=2
sync-binlog=1
binlog-ignore-db=information_schema
binlog-ignore-db=mysql
binlog-ignore-db=performance_schema
binlog-ignore-db=sys
#relay-log配置
relay_log=mysql-relay-bin
log_slave_updates=1
relay_log_purge=0
read_only=1
#gtid配置
#开启gtid
#gtid_mode=on
#enforce_gtid_consistency=1
```

重启服务

```
systemctl restart mysqld
```

开启同步

在Slave节点的MySQL命令行执行如下命令:

```
change master to
master_host='192.168.56.100',master_port=3306,master_user='root',master_passwo
rd='root',master_log_file='mysql-bin.000006',master_log_pos=154;

start slave; //开启同步
```

3.4 配置半同步复制

Master节点

安装插件

```
install plugin rpl_semi_sync_master soname 'semisync_master.so';
show variables like '%semi%';
```

修改配置文件

```
# 自动开启半同步复制
rpl_semi_sync_master_enabled=ON
rpl_semi_sync_master_timeout=1000
```

重启服务

```
systemctl restart mysqld
```

Slave节点

两台Slave节点都执行以下步骤：

安装插件

```
install plugin rpl_semi_sync_slave soname 'semisync_slave.so';
```

修改配置文件

```
# 自动开启半同步复制
rpl_semi_sync_slave_enabled=ON
```

重启服务

```
systemctl restart mysqld
```

测试半同步状态

首先通过MySQL命令行检查参数的方式，查看半同步是否开启。

```
show variables like '%semi%';
```

然后通过MySQL日志再次确认。

```
cat /var/log/mysqld.log
```

可以看到日志中已经启动半同步：

```
Start semi-sync replication to master 'root@192.168.56.100:3306' in log
'mysql-bin.000006' at position 154
```

4. MHA高可用搭建

四台机器ssh互通

在四台服务器上分别执行下面命令，生成公钥和私钥，换行回车采用默认值

```
ssh-keygen -t rsa
```

在三台MySQL服务器分别执行下面命令，讲公钥拷到MHA Manager服务器上

```
ssh-copy-id 192.168.56.103
```

之后可以在MHA Manager服务器上检查下，看看 .ssh/authorized_keys 文件是否包含3个公钥

```
cat /root/.ssh/authorized_keys
```

从MHA Manager服务器执行下面命令，向其他三台MySQL服务器分发公钥信息。

```
scp /root/.ssh/authorized_keys 192.168.56.100:$PWD
scp /root/.ssh/authorized_keys 192.168.56.101:$PWD
scp /root/.ssh/authorized_keys 192.168.56.102:$PWD
```

可以MHA Manager执行下面命令，检测下是否实现ssh互通。

```
ssh 192.168.56.100 // 或者ssh master100
```

MHA下载安装

MHA下载

MySQL5.7对应的MHA版本是0.5.8，所以在GitHub上找到对应的rpm包进行下载，MHA manager和node的安装包需要分别下载：

```
https://github.com/yoshinorim/mha4mysql-manager/releases/tag/v0.58
https://github.com/yoshinorim/mha4mysql-node/releases/tag/v0.58
```

下载后，将manager和node的安装包分别上传到对应的服务器 （可使用SCP等工具）。

```
scp /Users/liqiaoqiao/Downloads/mha4mysql-node-0.58-0.el7.centos.noarch.rpm
root@master100:
```

- 三台MySQL服务器需要安装node
- MHA Manager服务器需要安装manager和node

> 提示：也可以使用wget命令在linux系统直接下载获取，例如

```
wget https://github.com/yoshinorim/mha4mysql-
manager/releases/download/v0.58/mha4mysql-manager-0.58-
0.el7.centos.noarch.rpm
```

MHA node 安装

在四台服务器上安装mha4mysql-node。

MHA的Node依赖于perl-DBD-MySQL， 所以要先安装perl-DBD-MySQL。

```
yum install perl-DBD-MySQL -y
rpm -ivh mha4mysql-node-0.58-0.el7.centos.noarch.rpm
```

MHA manager 安装

在MHA Manager服务器安装mha4mysql-node和mha4mysql-manager。

MHA的manager又依赖了perl-Config-Tiny、perl-Log-Dispatch、perl-Parallel-ForkManager，也分别进行安装。

```
wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
rpm -ivh epel-release-latest-7.noarch.rpm
yum install perl-DBD-MySQL perl-Config-Tiny perl-Log-Dispatch perl-Parallel-
ForkManager -y

wget https://github.com/yoshinorim/mha4mysql-
node/releases/download/v0.58/mha4mysql-node-0.58-0.el7.centos.noarch.rpm
rpm -ivh mha4mysql-node-0.58-0.el7.centos.noarch.rpm

wget https://github.com/yoshinorim/mha4mysql-
manager/releases/download/v0.58/mha4mysql-manager-0.58-0.el7.centos.noarch.rpm
rpm -ivh mha4mysql-manager-0.58-0.el7.centos.noarch.rpm
```

> 提示：由于perl-Log-Dispatch和perl-Parallel-ForkManager这两个被依赖包在yum仓库找不到，因此安装epel-release-latest-7.noarch.rpm。在使用时，可能会出现下面异常：Cannot retrieve metalink for repository：epel/x86_64。可以尝试使用/etc/yum.repos.d/epel.repo，然后注释掉mirrorlist，取消注释baseurl。

MHA配置文件

MHA Manager服务器需要为每个监控的Master/Slave集群提供一个专用的配置文件，而所有的Master/Slave集群也可共享全局配置。

初始化配置目录

```
# 目录说明
# /var/log                          (CentOS目录)
#         /mha                      (MHA监控根目录)
#             /appl                 (MHA监控实例根目录)
#                  /manager.log     (MHA监控实例日志文件)
mkdir -p /var/log/mha/appl
touch /var/log/mha/appl/manager.log
```

配置监控全局配置文件

```
vi /etc/masterha_default.cnf
```

```
[server default]
user=root
password=root
ssh_user=root
repl_user=root
repl_password=root
ping_interval=1
secondary_check_script=masterha_secondary_check -s 192.168.56.100 -s
192.168.56.101 -s 192.168.56.102
```

配置监控实例配置文件

```
mkdir -p /etc/mha/
vi /etc/mha/appl.cnf
```

```
[server default]
manager_workdir=/var/log/mha/appl
manager_log=/var/log/mha/appl/manager.log

[server1]
hostname=192.168.56.100
candidate_master=1
master_binlog_dir="/var/lib/mysql"

[server2]
hostname=192.168.56.101
candidate_master=1
master_binlog_dir="/var/lib/mysql"

[server3]
hostname=192.168.56.102
candidate_master=1
master_binlog_dir="/var/lib/mysql"
```

MHA配置检测

执行ssh通信检测

在MHA Manager服务器上执行：

```
masterha_check_ssh --conf=/etc/mha/appl.cnf
```

检测MySQL主从复制

在MHA Manager服务器上执行：

```
masterha_check_repl --conf=/etc/mha/appl.cnf
```

出现“MySQL Replication Health is OK." 证明MySQL复制集群没有问题。

MHA Manager启动

在MHA Manager服务器上执行：

```
nohup masterha_manager --conf=/etc/mha/appl.cnf --remove_dead_master_conf --
ignore_last_failover < /dev/null > /var/log/mha/appl/manager.log 2>&1 &
```

查看监控状态命令如下：

```
masterha_check_status --conf=/etc/mha/appl.cnf
```

查看监控日志命令如下：

```
tail -f /var/log/mha/appl/manager.log
```

测试MHA故障转移

模拟主节点崩溃

在MHA Manager服务器执行打开日志命令：

```
tail -200f /var/log/masterha/appl/appl.log
```

关闭Master MySQL服务器服务，模拟主节点崩溃

```
systemctl stop mysqld
```

查看MHA日志，可以看到哪台slave切换成了master

```
show master status;
```

将原主节点切换回主

启动MySQL服务

```
systemctl start mysqld
```

挂到新主做从库

```
change master to
master_host='192.168.56.xxx',master_port=3306,master_user='root',master_passwo
rd='root', master_log_file='xxx',master_log_pos=xxx;
start slave;  //开启同步
```

> 提示：master_log_file和master_log_pos两个参数需要去新主库查看，show master status \G;

使用MHA在线切换命令将原主切换回来

```
masterha_master_switch --conf=/etc/mha/appl.cnf --master_state=alive --
new_master_host=192.168.56.100 --new_master_port=3306 --
orig_master_is_new_slave --running_updates_limit=10000
```

测试SQL脚本

```
create table position(
id int(20),
name varchar(50),
salary varchar(20),
city varchar(50)
) engine=innodb charset=utf8;
```

```
insert into position values(1, 'Java', 13000, 'shanghai');
insert into position values(2, 'DBA', 20000, 'beijing');
```

```
create table position_detail(
id int(20),
pid int(20),
description text
) engine=innodb charset=utf8;
```

```
insert into position_detail values(1, 1, 'Java Developer');
insert into position_detail values(2, 2, 'Database Administrator');
```