正文

一、简介

MHA（Master HA）是一款开源的 MySQL 的高可用程序，它为 MySQL 主从复制架构提供了 automating master failover 功能。MHA 在监控到 master 节点故障时，会提升其中拥有最新数据的 slave 节点成为新的master 节点，在此期间，MHA 会通过于其它从节点获取额外信息来避免一致性方面的问题。MHA 还提供了 master 节点的在线切换功能，即按需切换 master/slave 节点。

MHA 是由日本人 yoshinorim（原就职于DeNA现就职于FaceBook）开发的

比较成熟的 MySQL 高可用方案。MHA 能够在30秒内实现故障切换，并能在故障切换中，最大可能的保证数据一致性。目前淘宝也正在开发相似产品 TMHA， 目前已支持一主一从。

二、MHA 服务

2.1 服务角色

MHA 服务有两种角色， MHA Manager(管理节点)和 MHA Node(数据节点)：

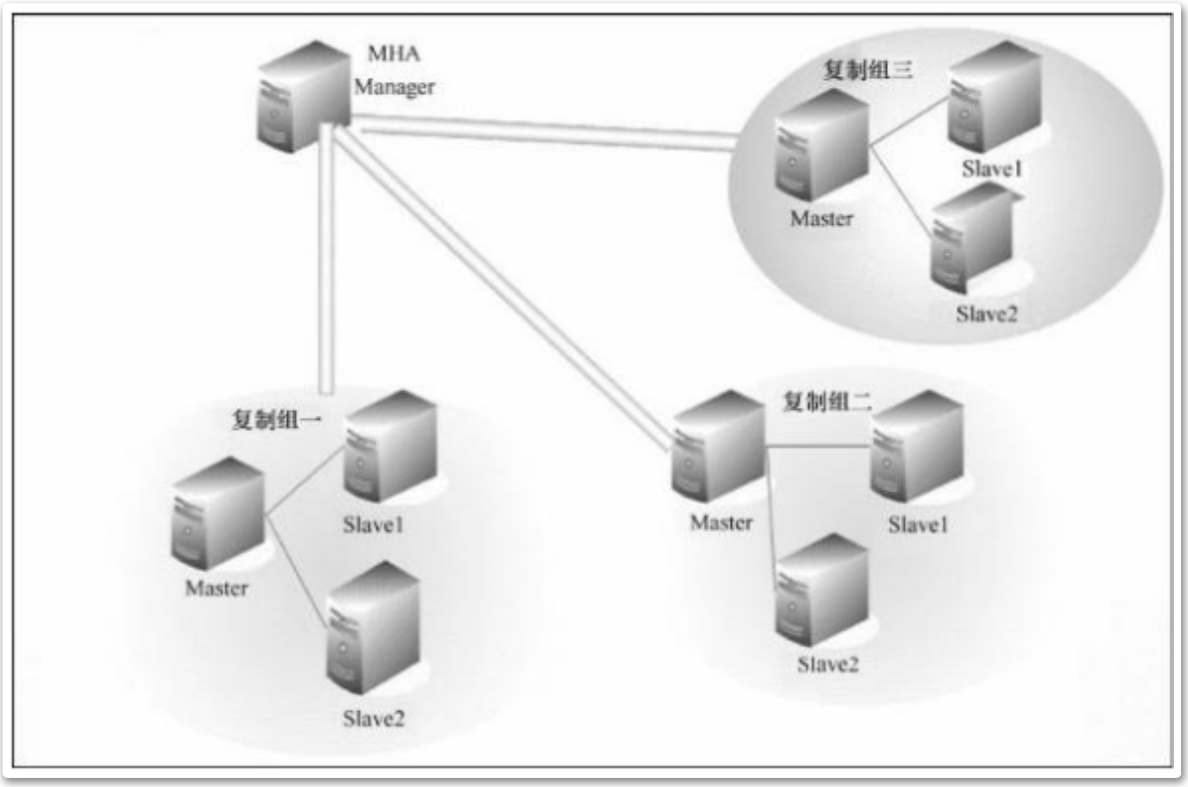
MHA Manager:

通常单独部署在一台独立机器上管理多个 master/slave 集群(组)，每个 master/slave 集群称作一个 **application**，用来管理统筹整个集群。

MHA node:

运行在每台 MySQL 服务器上(master/slave/manager)，它**通过监控具备解析和清理 logs 功能的脚本来加快故障转移。**

主要是接收管理节点所发出指令的代理，代理需要运行在每一个 mysql 节点上。简单讲 node 就是用来收集从节点服务器上所生成的 bin-log 。对比打算提升为新的主节点之上的从节点的是否拥有并完成操作，如果没有发给新主节点在本地应用后提升为主节点。



由上图我们可以看出，每个复制组内部和 Manager 之间都需要ssh实现无密码互连，只有这样，在 Master 出故障时， Manager 才能顺利的连接进去，实现主从切换功能。

2.2提供的工具

MHA会提供诸多工具程序， 其常见的如下所示：

Manager节点：

masterha_check_ssh：MHA 依赖的 ssh 环境监测工具；

masterha_check_repl：MYSQL 复制环境检测工具；

masterga_manager：MHA 服务主程序；

masterha_check_status：MHA 运行状态探测工具；

masterha_master_monitor：MYSQL master 节点可用性监测工具；

masterha_master_swith:master：节点切换工具；

masterha_conf_host：添加或删除配置的节点；

masterha_stop：关闭 MHA 服务的工具。

Node节点： （这些工具通常由MHA Manager的脚本触发，无需人为操作）

save_binary_logs：保存和复制 master 的二进制日志；

apply_diff_relay_logs：识别差异的中继日志事件并应用于其他 slave；

purge_relay_logs：清除中继日志（不会阻塞 SQL 线程）；

自定义扩展：

secondary_check_script：通过多条网络路由检测master的可用性；

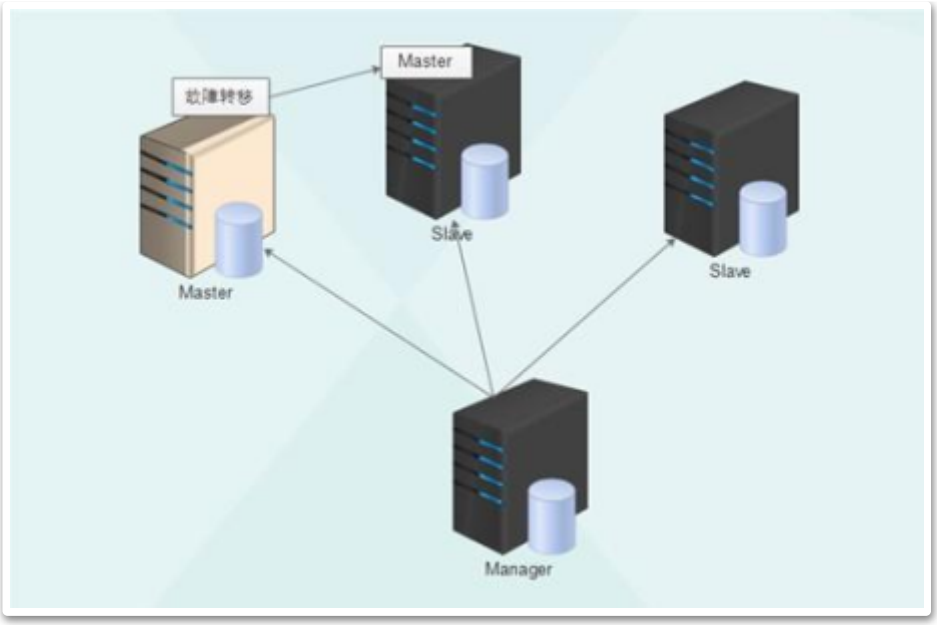
master_ip_failover_script：更新application使用的masterip；

report_script：发送报告；

init_conf_load_script：加载初始配置参数；

master_ip_online_change_script;更新master节点ip地址。

2.3工作原理



MHA工作原理总结为以下几条：

（1） 从宕机崩溃的 master 保存二进制日志事件（binlog events）；

（2） 识别含有最新更新的 slave；

（3） 应用差异的中继日志(relay log) 到其他 slave；

（4） 应用从 master 保存的二进制日志事件(binlog events)；

（5） 提升一个 slave 为新 master；

（6） 使用其他的 slave 连接新的 master 进行复制。

三、实现过程

3.1 准备实验 Mysql 的 Replication 环境

3.1.1 相关配置

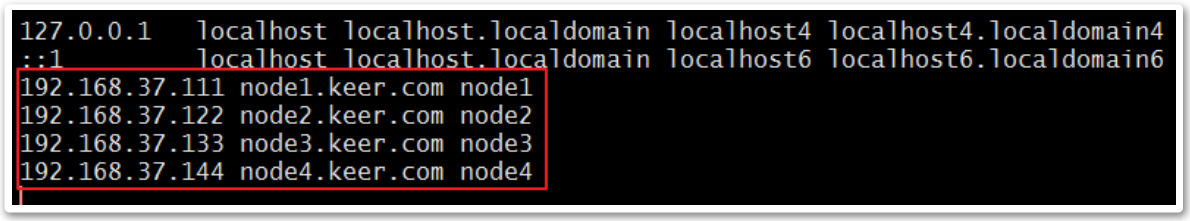
MHA 对 MYSQL 复制环境有特殊要求，例如各节点都要开启二进制日志及中继日志，各从节点必须显示启用其`read-only`属性，并关闭`relay_log_purge`功能等，这里对配置做事先说明。

本实验环境共有四个节点， 其角色分配如下（实验机器均为centos 7.3）：

机器名称	IP配置	服务角色	备注
Manager	192.168.37.111	Manager控制器	用于监控管理
master	192.168.37.122	数据库主服务器	开启bin-log relay-log 关闭relay_log_purge
slave1	192.168.37.133	数据库从服务器	开启bin-log relay-log 关闭relay_log_purge
slave2	192.168.37.144	数据库从服务器	开启bin-log relay-log 关闭relay_log_purge

为了方便我们后期的操作，我们在各节点的/etc/hosts文件配置内容中添加如下内容：

```
192.168.37.111 node1.keer.com node1
192.168.37.122 node2.keer.com node2
192.168.37.133 node3.keer.com node3
192.168.37.144 node4.keer.com node4
```



这样的话，我们就可以通过 host 解析节点来打通私钥访问，会方便很多。

本步骤完成。

3.1.2 初始主节点 master 的配置

我们需要修改 master 的数据库配置文件来对其进行初始化配置：

```
[root@master ~]# vim /etc/my.cnf
    [mysqld]
    server-id = 1                //复制集群中的各节点的id均必须唯一
    log-bin = master-log         //开启二进制日志
    relay-log = relay-log        //开启中继日志
    skip_name_resolve            //关闭名称解析（非必须）
[root@master ~]# systemctl restart mariadb
```

```
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
# Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0
# Settings user and group are ignored when systemd is used.
# If you need to run mysqld under a different user or group,
# customize your systemd unit file for mariadb according to the
# instructions in http://fedoraproject.org/wiki/Systemd

server-id = 1
log-bin = mysql-bin
skip-name-resolve
relay-log = mysql-relay-bin

[mysqld_safe]
log-error=/var/log/mariadb/mariadb.log
pid-file=/var/run/mariadb/mariadb.pid

#
# include all files from the config directory
#
!includedir /etc/my.cnf.d
```

本步骤完成。

3.1.3 所有 slave 节点依赖的配置

我们修改两个 slave 的数据库配置文件，两台机器都做如下操作：

```
[root@slave1 ~]# vim /etc/my.cnf
    [mysqld]
    server-id = 2                //复制集群中的各节点的id均必须唯一；
    relay-log = relay-log        //开启中继日志
    log-bin = master-log         //开启二进制日志
    read_only = ON               //启用只读属性
    relay_log_purge = 0          //是否自动清空不再需要中继日志
    skip_name_resolve            //关闭名称解析（非必须）
    log_slave_updates = 1        //使得更新的数据写进二进制日志中
[root@slave1 ~]# systemctl restart mariadb
[root@slave2 ~]# vim /etc/my.cnf
    [mysqld]
    server-id = 3                //复制集群中的各节点的id均必须唯一；
    relay-log = relay-log        //开启中继日志
    log-bin = master-log         //开启二进制日志
```

```
    read_only = ON                //启用只读属性
    relay_log_purge = 0           //是否自动清空不再需要中继日志
    skip_name_resolve             //关闭名称解析（非必须）
    log_slave_updates = 1         //使得更新的数据写进二进制日志中
[root@slave2 ~]# systemctl restart mariadb
```

本步骤完成。

3.1.4 配置一主多从复制架构

下面只会给出命令，具体的知识及过程详解就不再多说了。

master 节点上：

```
MariaDB [(none)]>grant replication slave,replication client on *.* to 'slave'@'192.168.%.%' identified by 'keer';
MariaDB [(none)]> show master status;
```

slave 节点上：

```
MariaDB [(none)]> change master to master_host='192.168.37.122',
    -> master_user='slave',
    -> master_password='keer',
    -> master_log_file='mysql-bin.000001',
    -> master_log_pos=415;
MariaDB [(none)]> start slave;
MariaDB [(none)]> show slave status\G;
```

本步骤完成。

3.2 安装配置MHA

3.2.1 在 master 上进行授权

在所有 Mysql 节点授权拥有管理权限的用户可在本地网络中有其他节点上远程访问。 当然， 此时仅需要且只能在 master 节点运行类似如下 SQL 语句即可。

```
MariaDB [(none)]> grant all on *.* to 'mhaadmin'@'192.168.%.%' identified by 'mhapass';
```

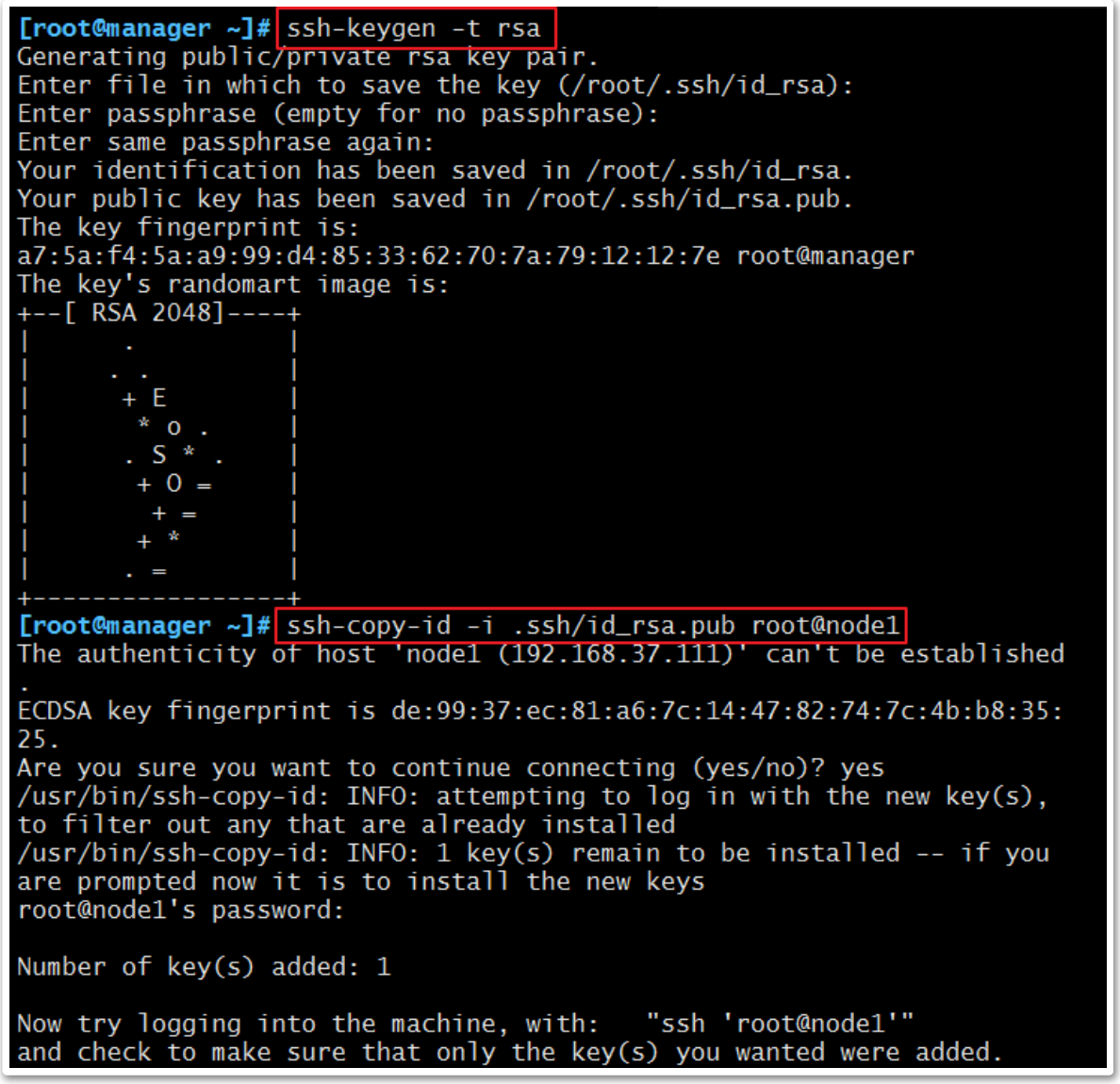
本步骤完成。

3.2.2 准备 ssh 互通环境

MHA集群中的各节点彼此之间均需要基于ssh互信通信，以实现远程控制及数据管理功能。简单起见，可在Manager节点生成密钥对儿，并设置其可远程连接本地主机后， 将私钥文件及authorized_keys文件复制给余下的所有节点即可。

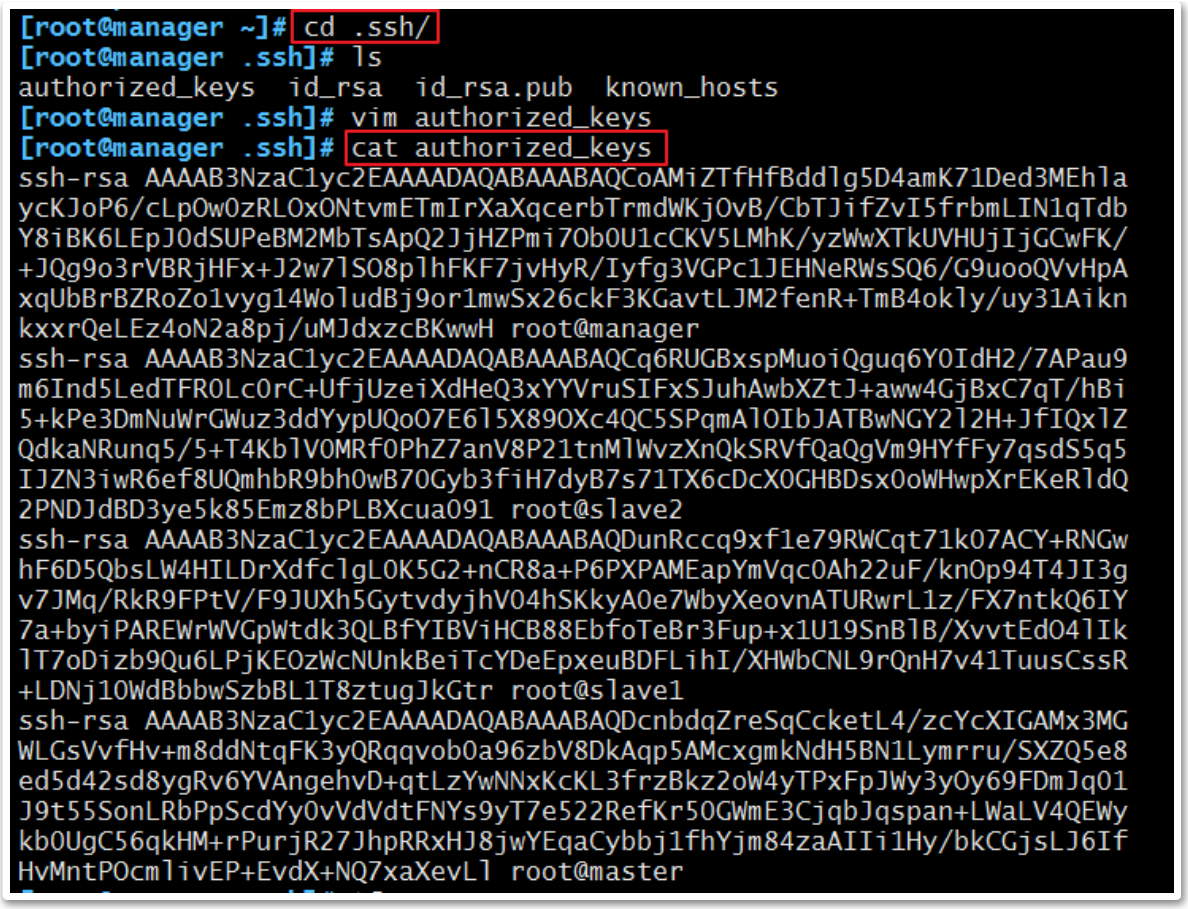
下面操作在所有节点上操作：

```
[root@manager ~]# ssh-keygen -t rsa
[root@manager ~]# ssh-copy-id -i .ssh/id_rsa.pub root@node1
```



当四台机器都进行了上述操作以后，我们可以在 manager 机器上看到如下文件：

```
[root@manager ~]# cd .ssh/
[root@manager .ssh]# ls
authorized_keys  id_rsa  id_rsa.pub  known_hosts
[root@manager .ssh]# cat authorized_keys
```



四台机器的公钥都已经在`authorized_keys`这个文件中了，接着，我们只需要把这个文件发送至另外三台机器，这四台机器就可以实现 ssh 无密码互通了：

```
[root@manager .ssh]# scp authorized_keys root@node2:~/.ssh/
[root@manager .ssh]# scp authorized_keys root@node3:~/.ssh/
[root@manager .ssh]# scp authorized_keys root@node4:~/.ssh/
```

当然，我们也可以在机器上实验一下，看看 ssh 是否还需要输入密码。

本步骤完成。

3.2.3 安装 MHA 包

在本步骤中， Manager节点需要另外多安装一个包。具体需要安装的内容如下：

> 四个节点都需安装：`mha4mysql-node-0.56-0.el6.norch.rpm`
> Manager 节点另需要安装：`mha4mysql-manager-0.56-0.el6.noarch.rpm`

我们使用`rz`命令分别上传，然后使用`yum`安装即可。

```
[root@manager ~]# rz
[root@manager ~]# ls
anaconda-ks.cfg  initial-setup-ks.cfg                    Pictures
Desktop          mha4mysql-manager-0.56-0.el6.noarch.rpm  Public
Documents        mha4mysql-node-0.56-0.el6.noarch.rpm     Templates
Downloads        Music                                    Videos
[root@manager ~]# yum install -y mha4mysql-node-0.56-0.el6.noarch.rpm
[root@manager ~]# yum install -y mha4mysql-manager-0.56-0.el6.noarch.rpm
```

其余机器也分别进行安装，就不一一举例了。

本步骤完成。

3.2.4 初始化 MHA，进行配置

Manager 节点需要为每个监控的 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**。全局配置文件**默认为**`/etc/masterha_default.cnf`，其为**可选配置**。如果仅监控一组 master/slave 集群，也可直接通过 application 的配置来提供各服务器的默认配置信息。而每个 application 的配置文件路径为自定义。具体操作见下一步骤。

3.2.5 定义 MHA 管理配置文件

为MHA专门创建一个管理用户， 方便以后使用， 在mysql的主节点上， 三个节点自动同步：

```
    mkdir /etc/mha_master
    vim /etc/mha_master/mha.cnf
```

配置文件内容如下；

```
[server default]                //适用于server1,2,3个server的配置
user=mhaadmin                   //mha管理用户
password=mhapass                //mha管理密码
manager_workdir=/etc/mha_master/app1          //mha_master自己的工作路径
manager_log=/etc/mha_master/manager.log       // mha_master自己的日志文件
remote_workdir=/mydata/mha_master/app1        //每个远程主机的工作目录在何处
ssh_user=root                   // 基于ssh的密钥认证
repl_user=slave                 //数据库用户名
repl_password=magedu            //数据库密码
ping_interval=1                 //ping间隔时长
[server1]                       //节点2
hostname=192.168.37.133         //节点2主机地址
```

```
ssh_port=22                    //节点2的ssh端口
candidate_master=1             //将来可不可以成为master候选节点/主节点
[server2]
hostname=192.168.37.133
ssh_port=22
candidate_master=1
[server3]
hostname=192.168.37.144
ssh_port=22
candidate_master=1
```

本步骤完成。

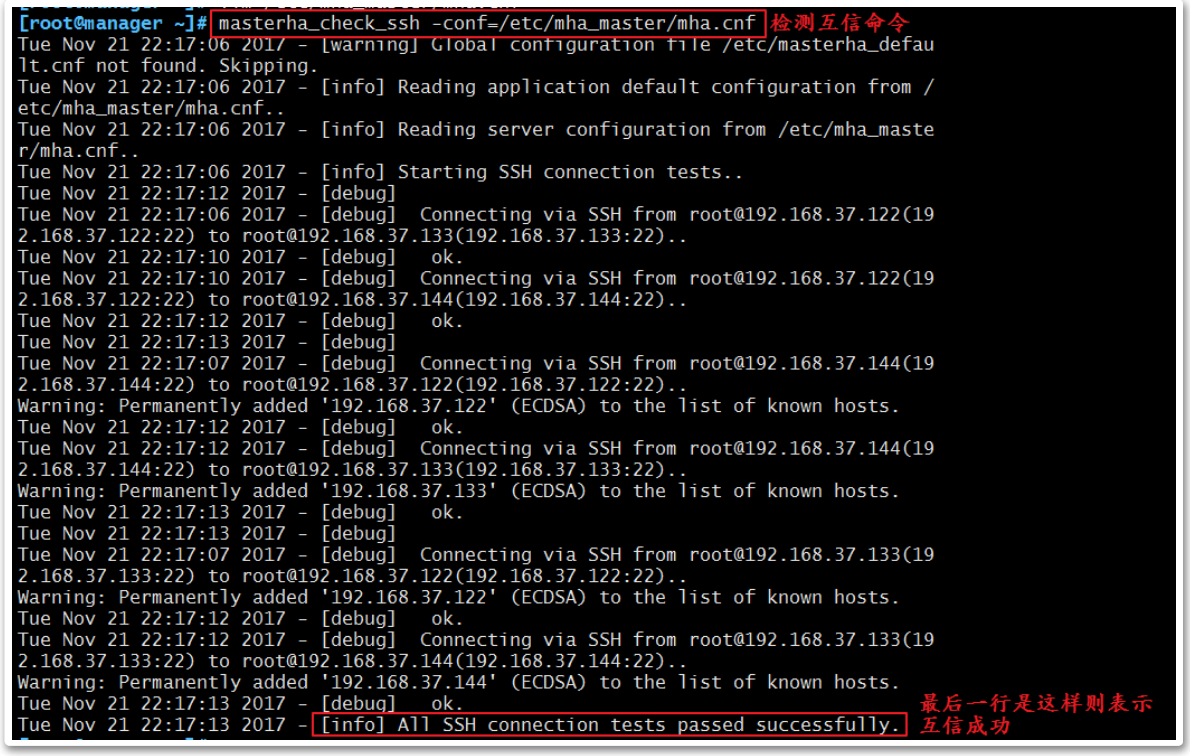
3.2.6 对四个节点进行检测

1）检测各节点间 ssh 互信通信配置是否 ok

我们在 Manager 机器上输入下述命令来检测：

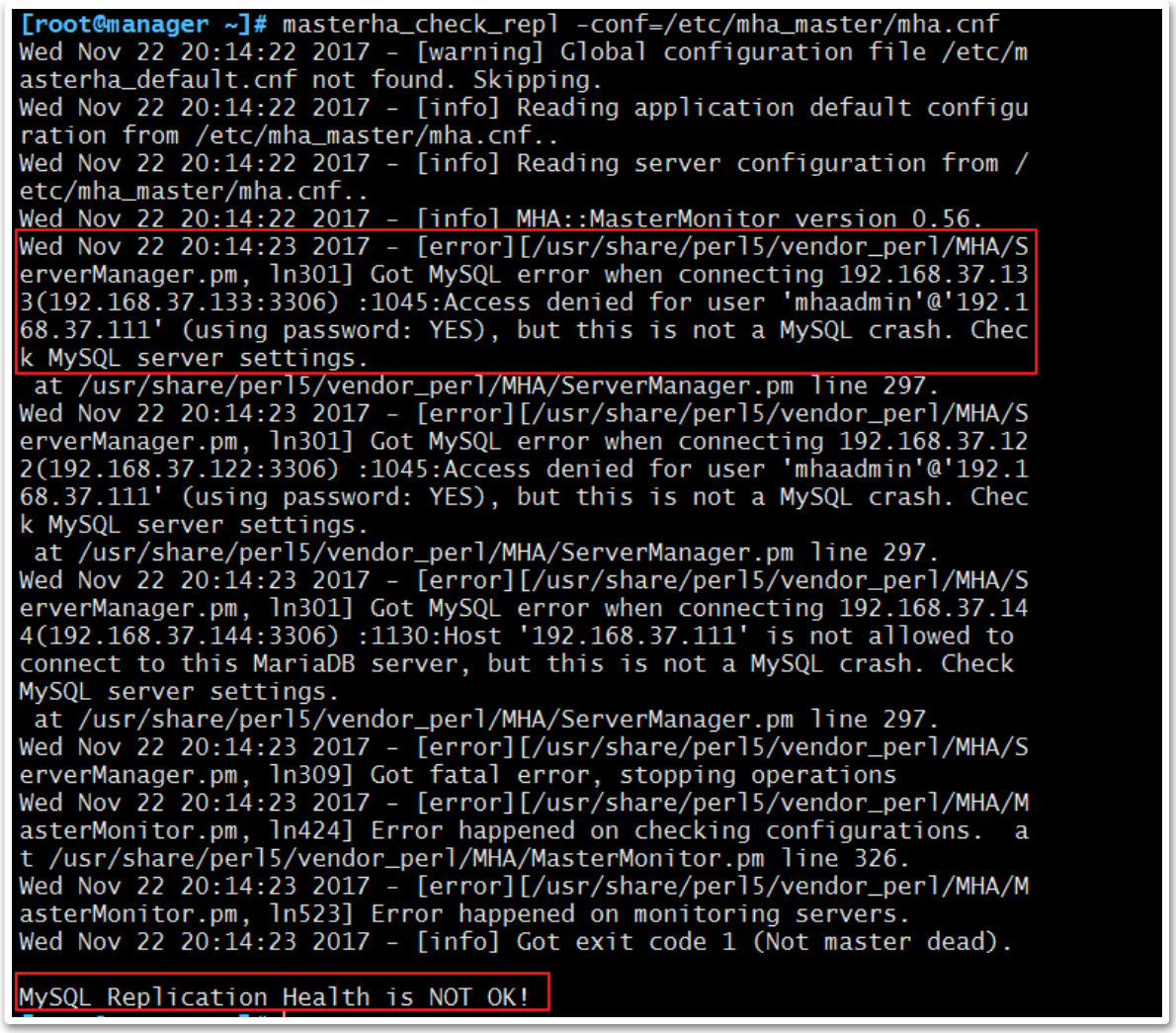
```
[root@manager ~]# masterha_check_ssh -conf=/etc/mha_master/mha.cnf
```

如果最后一行显示为`[info]All SSH connection tests passed successfully.`则表示成功。



2）检查管理的MySQL复制集群的连接配置参数是否OK

```
[root@manager ~]# masterha_check_repl -conf=/etc/mha_master/mha.cnf
```



我们发现检测失败，这可能是因为从节点上没有账号，因为这个架构，任何一个从节点， 将有可能成为主节点， 所以也需要创建账号。

因此，我们需要在master节点上再次执行以下操作：

```
MariaDB [(none)]> grant replication slave,replication client on *.* to 'slave'@'192.168.%.%' identified by 'keer';
MariaDB [(none)]> flush privileges;
```

执行完这段操作之后，我们再次运行检测命令：

```
[root@manager ~]# masterha_check_repl -conf=/etc/mha_master/mha.cnf
Thu Nov 23 09:07:08 2017 - [warning] Global configuration file /etc/masterha_default.cnf not found. Skipping.
Thu Nov 23 09:07:08 2017 - [info] Reading application default configuration from /etc/mha_master/mha.cnf..
Thu Nov 23 09:07:08 2017 - [info] Reading server configuration from /etc/mha_master/mha.cnf..
……
MySQL Replication Health is OK.
```

可以看出，我们的检测已经ok了。

此步骤完成。

3.3 启动 MHA

我们在 manager 节点上执行以下命令来启动 MHA：

```
[root@manager ~]# nohup masterha_manager -conf=/etc/mha_master/mha.cnf &>
/etc/mha_master/manager.log &
[1] 7598
```

启动成功以后，我们来查看一下 master 节点的状态：

```
[root@manager ~]# masterha_check_status -conf=/etc/mha_master/mha.cnf
mha (pid:7598) is running(0:PING_OK), master:192.168.37.122
```

上面的信息中“mha (pid:7598) is running(0:PING_OK)”表示MHA服务运行OK，否则， 则会显示为类似“mha is stopped(1:NOT_RUNNING).”

如果，我们想要停止 MHA ，则需要使用 stop 命令：

```
[root@manager ~]# masterha_stop -conf=/etc/mha_master/mha.cnf
```

3.4 测试 MHA 故障转移

3.4.1 在 master 节点关闭 mariadb 服务,模拟主节点数据崩溃

```
[root@master ~]# killall -9 mysqld mysqld_safe
[root@master ~]# rm -rf /var/lib/mysql/*
```

3.4.2 在 manger 节点查看日志

我们来查看日志：

```
[root@manager ~]# tail -200 /etc/mha_master/manager.log
……
Thu Nov 23 09:17:19 2017 - [info] Master failover to
192.168.37.133(192.168.37.133:3306) completed successfully.
```

表示 manager 检测到192.168.37.122节点故障， 而后自动执行故障转移， 将192.168.37.133提升为主节点。

注意，**故障转移完成后， manager将会自动停止**， 此时使用masterha_check_status 命令检测将会遇到错误提示， 如下所示：

```
[root@manager ~]# masterha_check_status -conf=/etc/mha_master/mha.cnf
mha is stopped(2:NOT_RUNNING).
```

3.5 提供新的从节点以修复复制集群

原有 master 节点故障后，需要重新准备好一个新的 MySQL 节点。基于来自于master 节点的备份恢复数据后，将其配置为新的 master 的从节点即可。注意，新加入的节点如果为新增节点，其 IP 地址要配置为原来 master 节点的 IP，否则，还需要修改 mha.cnf 中相应的 ip 地址。随后再次启动 manager ，并再次检测其状态。

我们就以刚刚关闭的那台主作为新添加的机器，来进行数据库的恢复：

原本的 slave1 已经成为了新的主机器，所以，我们对其进行完全备份，而后把备份的数据发送到我们新添加的机器上：

```
[root@slave1 ~]# mkdir /backup
[root@slave1 ~]# mysqldump --all-database > /backup/mysql-backup-`date +%F-%T`-all.sql
[root@slave1 ~]# scp /backup/mysql-backup-2017-11-23-09\:57\:09-all.sql root@node2:~
```

然后在 node2 节点上进行数据恢复：

```
[root@master ~]# mysql < mysql-backup-2017-11-23-09\:57\:09-all.sql
```

接下来就是配置主从。照例查看一下现在的主的二进制日志和位置，然后就进行如下设置：

```
MariaDB [(none)]> change master to master_host='192.168.37.133', master_user='slave',  master_password='keer', master_log_file='mysql-bin.000006', master_log_pos=925;
MariaDB [(none)]> start slave;
MariaDB [(none)]> show slave status\G;
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 192.168.37.133
```

```
                  Master_User: slave
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: mysql-bin.000006
          Read_Master_Log_Pos: 925
               Relay_Log_File: mysql-relay-bin.000002
                Relay_Log_Pos: 529
        Relay_Master_Log_File: mysql-bin.000006
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
                              ……
```

可以看出，我们的主从已经配置好了。

本步骤完成。

3.6 新节点提供后再次执行检查操作

我们来再次检测状态：

```
[root@manager ~]# masterha_check_repl -conf=/etc/mha_master/mha.cnf
```

如果报错，则再次授权。若没有问题，则启动 manager，注意，这次启动要**记录日志**：

```
[root@manager ~]# masterha_manager -conf=/etc/mha_master/mha.cnf >
/etc/mha_master/manager.log 2>&1 &
[1] 10012
```

启动成功以后，我们来查看一下 master 节点的状态：

```
[root@manager ~]# masterha_check_status -conf=/etc/mha_master/mha.cnf
mha (pid:9561) is running(0:PING_OK), master:192.168.37.133
```

我们的服务已经成功继续了。

本步骤结束。

3.7新节点上线， 故障转换恢复注意事项

1）在生产环境中， 当你的主节点挂了后， 一定要在从节点上做一个备份， 拿着备份文件把主节点手动提升为从节点， 并指明从哪一个日志文件的位置开始复制

2）每一次自动完成转换后， 每一次的(replication health)检测不ok始终都是启动不了必须手动修复主节点， 除非你改配置文件

3）手动修复主节点提升为从节点后， 再次运行检测命令

```
[root@manager ~]# masterha_check_status -conf=/etc/mha_master/mha.cnf
mha (pid:9561) is running(0:PING_OK), master:192.168.37.133
```

4）再次运行起来就恢复成功了

```
[root@manager ~]# masterha_manager --conf=/etc/mha_master/mha.cnf
```

以上。我们的实验已经圆满完成。