126 数据库高可用：基于主从复制实现故障转移（2）

今天我们正式来讲解MHA数据库高可用架构的搭建，先来讲解一下在三个数据库所在机器上安装MHA node节点的步骤，首先那必须要先安装Perl语言环境了，这就跟我们平时用Java开发，那你必须得先装个JDK吧！

所以先可以用yum装一下Perl语言环境：yum install perl-DBD-MySQL

然后从下述地址下载MHA node代码：https://github.com/yoshinorim/mha4mysql-node，接着就可以把node的压缩包用WinSCP之类的工具上传到机器上去，接着解压缩node包就可以了，tar -zxvf mha4mysql-node-0.57.tar.gz。

然后可以安装perl-cpan软件包：

```
cd mha4mysql-node-0.57

yum -y install perl-CPAN*

perl Makefile.PL

make && make install
```

到此为止，暂时node的安装就可以了，记得3个部署MySQL的机器都要安装node，接着就是安装MHA的manager节点，先安装需要的一些依赖包：

```
yum install -y perl-DBD-MySQL*

rpm -ivh perl-Params-Validate-0.92-3.el6.x86_64.rpm

rpm -ivh perl-Config-Tiny-2.12-1.el6.rfx.noarch.rpm

rpm -ivh perl-Log-Dispatch-2.26-1.el6.rf.noarch.rpm

rpm -ivh perl-Parallel-ForkManager-0.7.5-2.2.el6.rf.noarch.rpm
```

接着就可以安装manager节点了，先在下面的地址下载manager的压缩包：https://github.com/yoshinorim/mha4mysql-manager，然后上传到机器上去，按照下述步骤安装就可以了：

```
tar -zxvf mha4mysql-manager-0.57.tar.gz

perl Makefile.PL

make
```

```
make install
```

接着为MHA manager创建几个目录：/usr/local/mha，/etc/mha，然后进入到/etc/mha目录下，vi mha.conf一下，编辑他的配合文件

```
[server default]
user=zhss
password=12345678
manager_workdir=/usr/local/mha
manager_log=/usr/local/mha/manager.log
remote_workdir=/usr/local/mha
ssh_user=root
repl_user=repl
repl_password=repl
ping_interval=1
master_ip_failover_script=/usr/local/scripts/master_ip_failover
master_ip_online_change_script=/usr/local/scripts/master_ip_online_change

[server1]
hostname=xx.xx.xx.xx
ssh_port=22
master_binlog_dir=/data/mysqll
condidate_master=1
port=3306

[server1]
hostname=xx.xx.xx.xx
ssh_port=22
master_binlog_dir=/data/mysqll
condidate_master=1
port=3306

[server1]
hostname=xx.xx.xx.xx
ssh_port=22
```

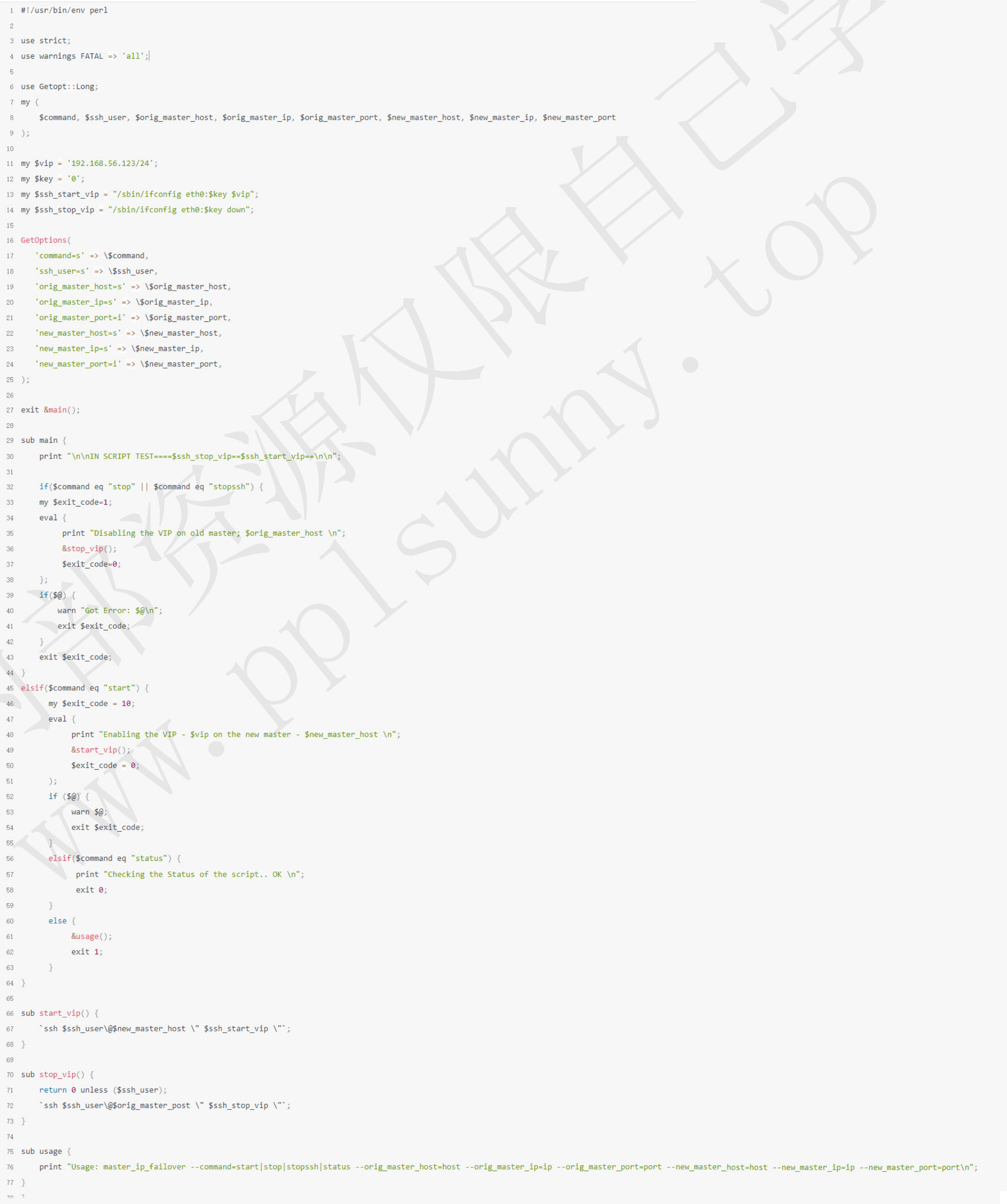
master_binlog_dir=/data/mysqll

condidate_master=1

port=3306

上面那份配置文件就可以指导MHA manager节点去跟其他节点的node通信了，大家可以观察到，上面说白了都是配置一些工作目录，日志目录，用户密码之类的东西，还有一些脚本，另外比较关键的是，你有几个node节点，就配置一个server，把每个server的ip地址配置进去就可以了

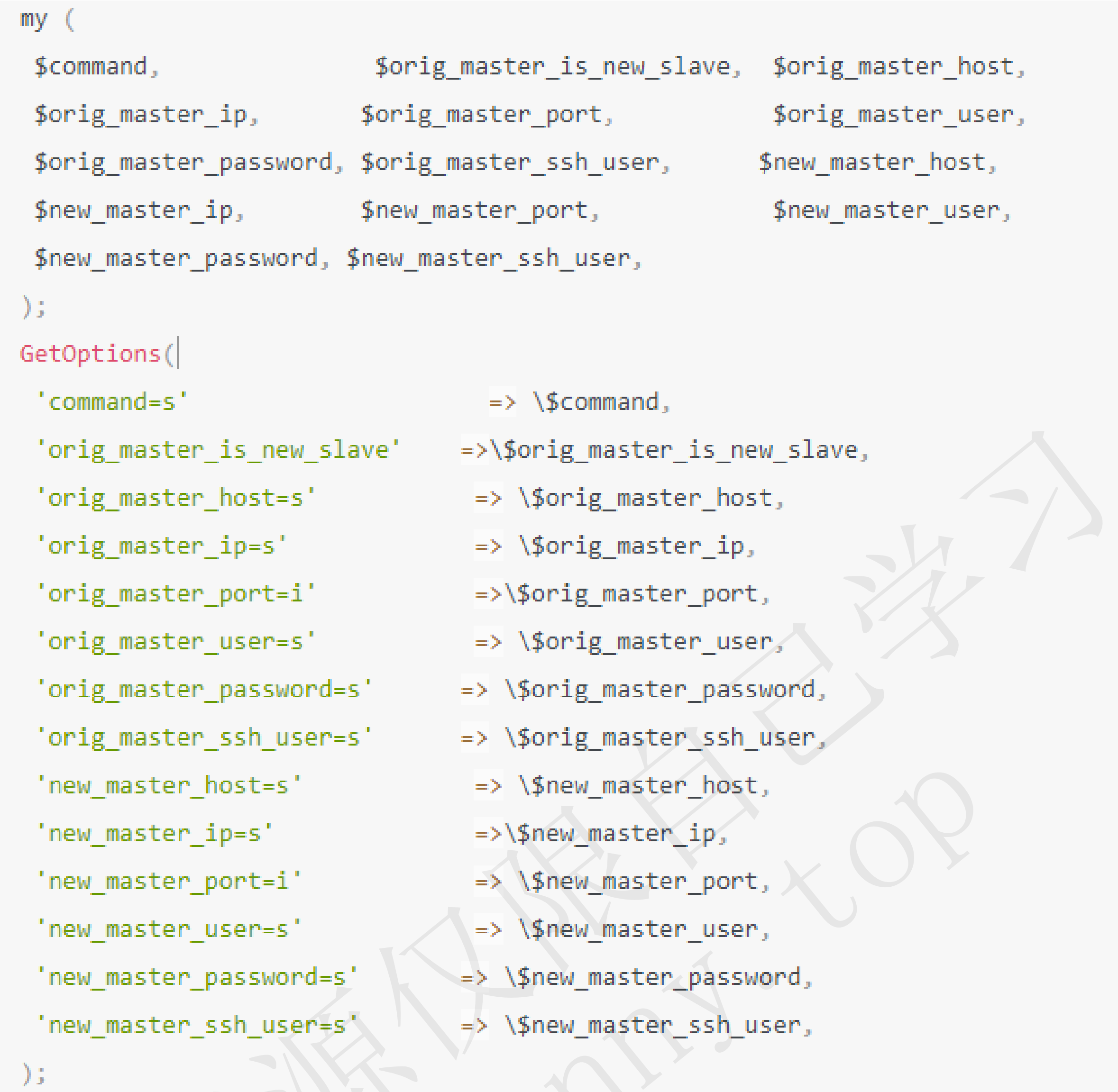
接着创建存放脚本的目录：/usr/local/scripts，在里面需要放一个master_ip_failover脚本，vi master_ip_failover就可以了，输入下面的内容：



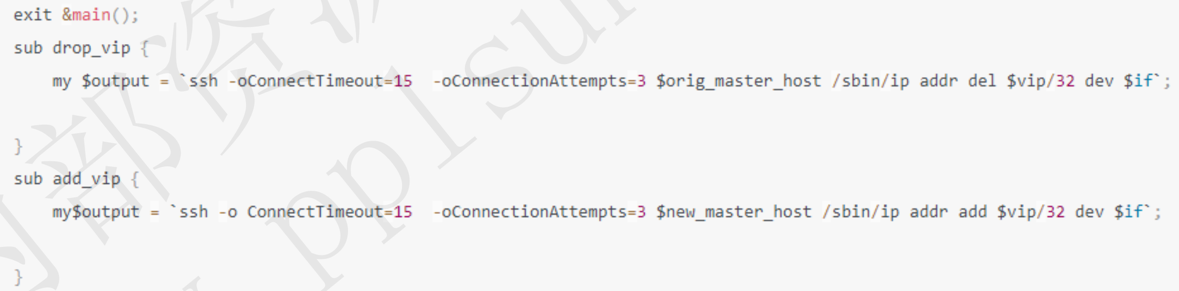
```
#!/usr/bin/env perl

use strict;
use warnings FATAL => 'all';

use Getopt::Long;
my (
    $command, $ssh_user, $orig_master_host, $orig_master_ip, $orig_master_port, $new_master_host, $new_master_ip, $new_master_port
);

my $vip = '192.168.56.123/24';
my $key = '0';
my $ssh_start_vip = "/sbin/ifconfig eth0:$key $vip";
my $ssh_stop_vip = "/sbin/ifconfig eth0:$key down";

GetOptions(
   'command=s' => \$command,
   'ssh_user=s' => \$ssh_user,
   'orig_master_host=s' => \$orig_master_host,
   'orig_master_ip=s' => \$orig_master_ip,
   'orig_master_port=i' => \$orig_master_port,
   'new_master_host=s' => \$new_master_host,
   'new_master_ip=s' => \$new_master_ip,
   'new_master_port=i' => \$new_master_port,
);

exit &main();

sub main {
    print "\n\nIN SCRIPT TEST====$ssh_stop_vip==$ssh_start_vip==\n\n";

    if($command eq "stop" || $command eq "stopssh") {
    my $exit_code=1;
    eval {
         print "Disabling the VIP on old master; $orig_master_host \n";
         &stop_vip();
         $exit_code=0;
    };
    if($@) {
        warn "Got Error: $@\n";
        exit $exit_code;
    }
    exit $exit_code;
}
elsif($command eq "start") {
      my $exit_code = 10;
      eval {
           print "Enabling the VIP - $vip on the new master - $new_master_host \n";
           &start_vip();
           $exit_code = 0;
      );
      if ($@) {
           warn $@;
           exit $exit_code;
      }
      elsif($command eq "status") {
            print "Checking the Status of the script.. OK \n";
            exit 0;
      }
      else {
           &usage();
           exit 1;
      }
}

sub start_vip() {
    `ssh $ssh_user\@$new_master_host \" $ssh_start_vip \"`;
}

sub stop_vip() {
    return 0 unless ($ssh_user);
    `ssh $ssh_user\@$orig_master_post \" $ssh_stop_vip \"`;
}

sub usage {
    print "Usage: master_ip_failover --command=start|stop|stopssh|status --orig_master_host=host --orig_master_ip=ip --orig_master_port=port --new_master_host=host --new_master_ip=ip --new_master_port=port\n";
}
}
```

接着在编辑一下online_change这个脚本，如下：

```
cd /usr/local/scripts/
vim master_ip_online_change
#!/usr/bin/env perl
use strict;
use warnings FATAL => 'all';

use Getopt::Long;
use MHA::DBHelper;
use MHA::NodeUtil;
use Time::HiRes qw( sleep gettimeofday tv_interval );
use Data::Dumper;

my $_tstart;
my $_running_interval = 0.1;

my $vip = "192.168.56.123";
my $if = "eth0";
```



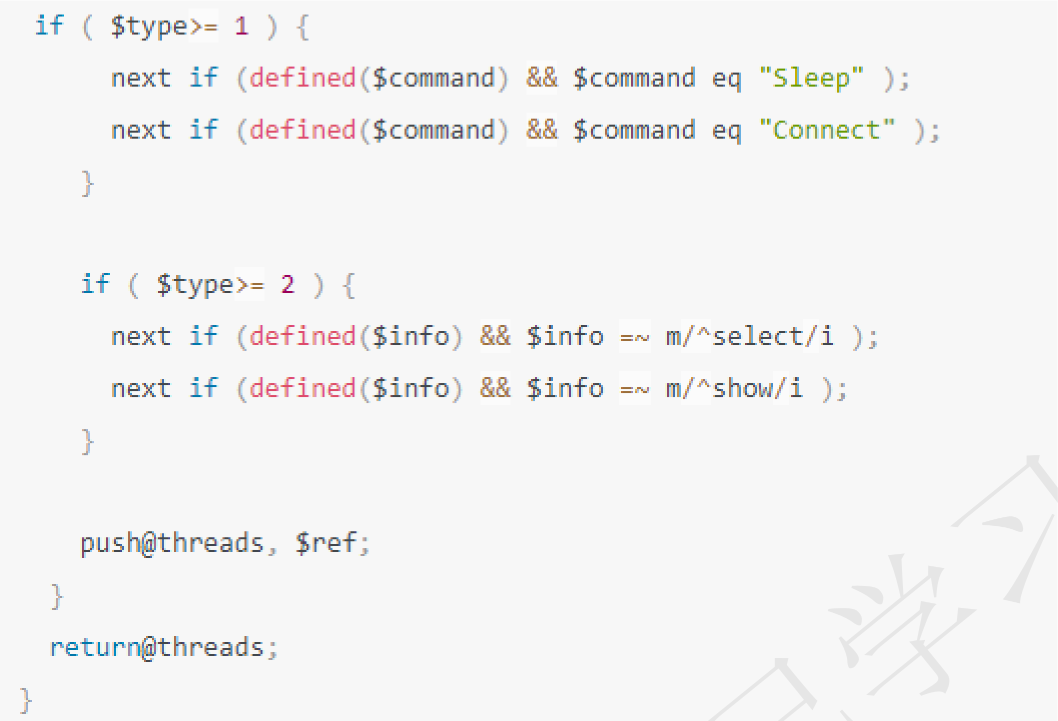
```
my (
 $command,                $orig_master_is_new_slave, $orig_master_host,
 $orig_master_ip,       $orig_master_port,           $orig_master_user,
 $orig_master_password, $orig_master_ssh_user,      $new_master_host,
 $new_master_ip,        $new_master_port,            $new_master_user,
 $new_master_password, $new_master_ssh_user,
);
GetOptions(
 'command=s'                     => \$command,
 'orig_master_is_new_slave'    =>\$orig_master_is_new_slave,
 'orig_master_host=s'           => \$orig_master_host,
 'orig_master_ip=s'             => \$orig_master_ip,
 'orig_master_port=i'           =>\$orig_master_port,
 'orig_master_user=s'           => \$orig_master_user,
 'orig_master_password=s'      => \$orig_master_password,
 'orig_master_ssh_user=s'      => \$orig_master_ssh_user,
 'new_master_host=s'            => \$new_master_host,
 'new_master_ip=s'              =>\$new_master_ip,
 'new_master_port=i'            => \$new_master_port,
 'new_master_user=s'            => \$new_master_user,
 'new_master_password=s'       => \$new_master_password,
 'new_master_ssh_user=s'       => \$new_master_ssh_user,
);
```



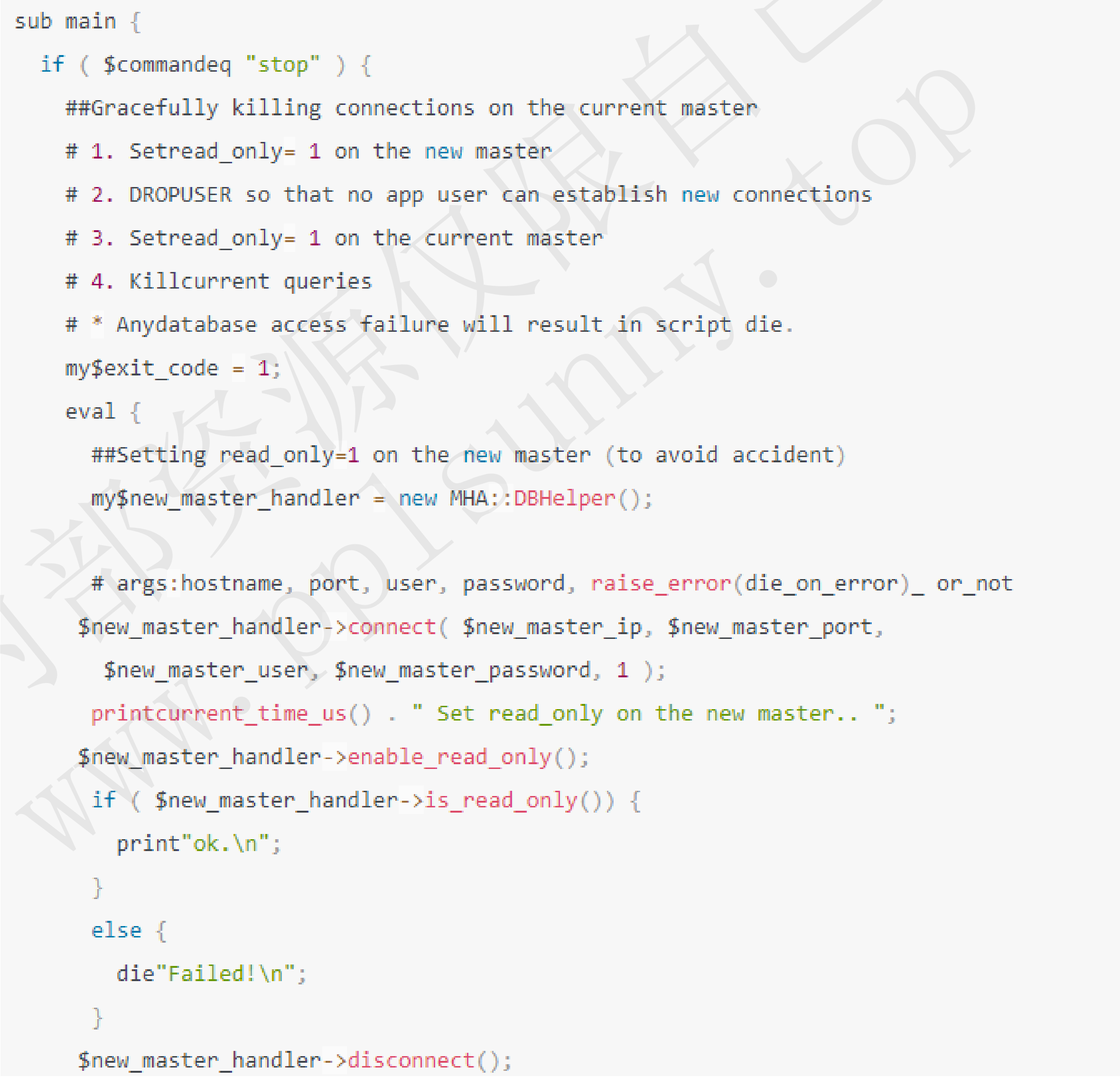
```
exit &main();
sub drop_vip {
    my $output = `ssh -oConnectTimeout=15  -oConnectionAttempts=3 $orig_master_host /sbin/ip addr del $vip/32 dev $if`;

}
sub add_vip {
    my$output = `ssh -o ConnectTimeout=15  -oConnectionAttempts=3 $new_master_host /sbin/ip addr add $vip/32 dev $if`;

}
```

```
sub current_time_us {
 my ( $sec,$microsec ) = gettimeofday();
 my $curdate =localtime($sec);
 return$curdate . " " . sprintf( "%06d", $microsec );
}

sub sleep_until {
 my $elapsed =tv_interval($_tstart);
 if ($_running_interval > $elapsed ) {
    sleep($_running_interval - $elapsed );
 }
}
```

```
sub get_threads_util {
 my $dbh                     = shift;
 my$my_connection_id        = shift;
 my$running_time_threshold = shift;
 my $type                    = shift;
 $running_time_threshold = 0 unless ($running_time_threshold);
  $type                    = 0 unless ($type);
  my @threads;

  my $sth =$dbh->prepare("SHOW PROCESSLIST");
 $sth->execute();

  while ( my$ref = $sth->fetchrow_hashref() ) {
   my $id          = $ref->{Id};
   my$user        = $ref->{User};
   my$host        = $ref->{Host};
   my$command     = $ref->{Command};
   my$state       = $ref->{State};
   my$query_time = $ref->{Time};
   my$info        = $ref->{Info};
   $info =~s/^\s*(.*?)\s*$/$1/ if defined($info);
   next if ($my_connection_id == $id );
   next if ( defined($query_time) &&$query_time < $running_time_threshold );
   next if (defined($command)    && $commandeq "Binlog Dump" );
   next if ( defined($user)       && $user eq "systemuser" );
   next
    if (defined($command)
    && $command eq "Sleep"
    && defined($query_time)
    && $query_time >= 1 );
```



```
  if ( $type>= 1 ) {
       next if (defined($command) && $command eq "Sleep" );
       next if (defined($command) && $command eq "Connect" );
     }

     if ( $type>= 2 ) {
       next if (defined($info) && $info =~ m/^select/i );
       next if (defined($info) && $info =~ m/^show/i );
     }

     push@threads, $ref;
   }
   return@threads;
 }
```



```
sub main {
  if ( $commandeq "stop" ) {
    ##Gracefully killing connections on the current master
    # 1. Setread_only= 1 on the new master
    # 2. DROPUSER so that no app user can establish new connections
    # 3. Setread_only= 1 on the current master
    # 4. Killcurrent queries
    # * Anydatabase access failure will result in script die.
    my$exit_code = 1;
    eval {
      ##Setting read_only=1 on the new master (to avoid accident)
      my$new_master_handler = new MHA::DBHelper();

      # args:hostname, port, user, password, raise_error(die_on_error)_ or_not
     $new_master_handler->connect( $new_master_ip, $new_master_port,
       $new_master_user, $new_master_password, 1 );
      printcurrent_time_us() . " Set read_only on the new master.. ";
     $new_master_handler->enable_read_only();
      if ( $new_master_handler->is_read_only()) {
        print"ok.\n";
      }
      else {
        die"Failed!\n";
      }
     $new_master_handler->disconnect();
```

```
#Connecting to the orig master, die if any database error happens
my$orig_master_handler = new MHA::DBHelper();
$orig_master_handler->connect( $orig_master_ip, $orig_master_port,
 $orig_master_user, $orig_master_password, 1 );

 ## Dropapplication user so that nobody can connect. Disabling per-session binlogbeforehand
 $orig_master_handler->disable_log_bin_local();
 # printcurrent_time_us() . " Drpping app user on the orig master..\n";
  printcurrent_time_us() . " drop vip $vip..\n";
 #drop_app_user($orig_master_handler);
 &drop_vip();
```

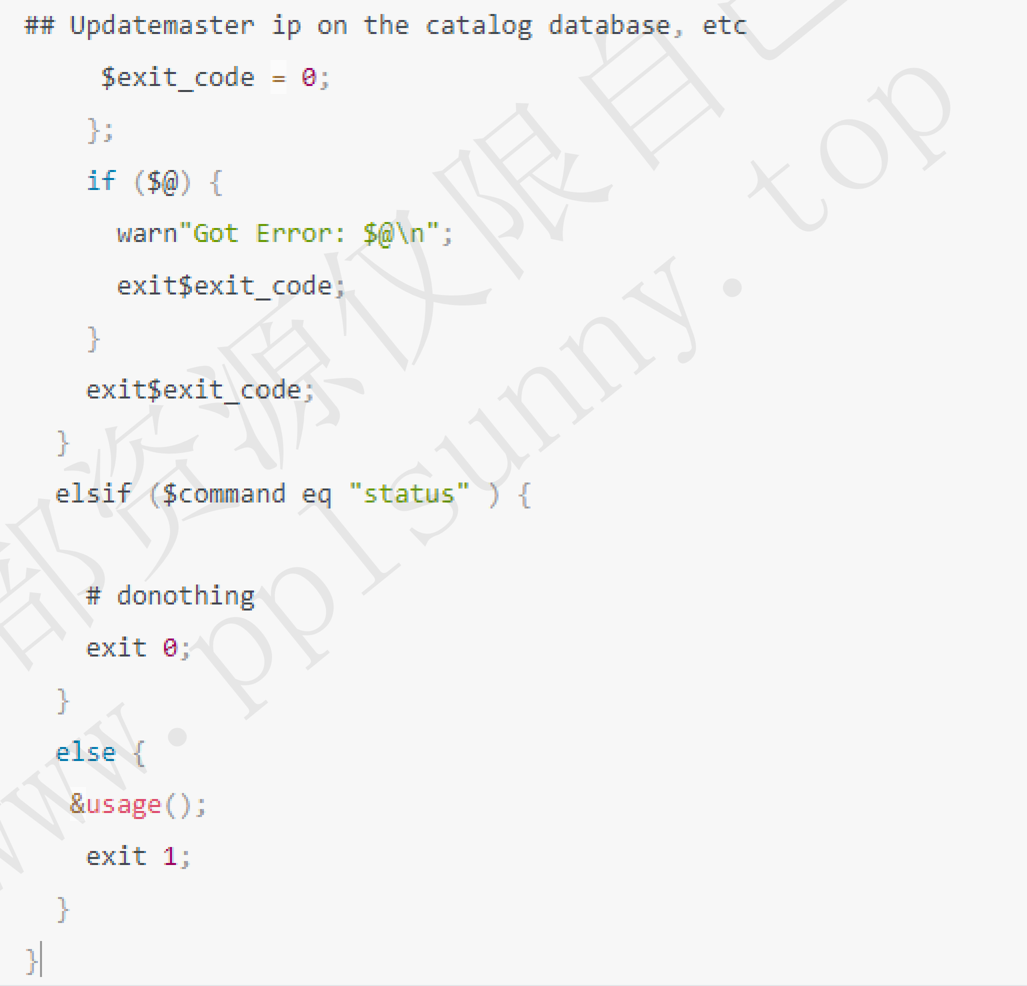
```
 ## Waitingfor N * 100 milliseconds so that current connections can exit
      my$time_until_read_only = 15;
      $_tstart= [gettimeofday];
      my@threads = get_threads_util( $orig_master_handler->{dbh},
       $orig_master_handler->{connection_id} );
      while ($time_until_read_only > 0 && $#threads >= 0 ) {
        if ($time_until_read_only % 5 == 0 ) {
         printf
"%s Waiting all running %d threads aredisconnected.. (max %d milliseconds)\n",
           current_time_us(), $#threads + 1, $time_until_read_only * 100;
          if ($#threads < 5 ) {
           print Data::Dumper->new( [$_] )->Indent(0)->Terse(1)->Dump ."\n"
              foreach (@threads);
          }
        }
```

```
sleep_until();
     $_tstart = [gettimeofday];
     $time_until_read_only--;
     @threads = get_threads_util( $orig_master_handler->{dbh},
       $orig_master_handler->{connection_id} );
     }

     ##Setting read_only=1 on the current master so that nobody(except SUPER) canwrite
     printcurrent_time_us() . " Set read_only=1 on the orig master.. ";
    $orig_master_handler->enable_read_only();
     if ($orig_master_handler->is_read_only() ) {
       print"ok.\n";
     }
     else {
       die"Failed!\n";
     }
```

```
##Waiting for M * 100 milliseconds so that current update queries can complete
      my$time_until_kill_threads = 5;
      @threads= get_threads_util( $orig_master_handler->{dbh},
       $orig_master_handler->{connection_id} );
      while ($time_until_kill_threads > 0 && $#threads >= 0 ) {
        if ($time_until_kill_threads % 5 == 0 ) {
         printf
"%sWaiting all running %d queries are disconnected.. (max %dmilliseconds)\n",
           current_time_us(), $#threads + 1, $time_until_kill_threads * 100;
          if ($#threads < 5 ) {
           print Data::Dumper->new( [$_] )->Indent(0)->Terse(1)->Dump ."\n"
             foreach (@threads);
          }
        }
       sleep_until();
       $_tstart = [gettimeofday];
        $time_until_kill_threads--;
       @threads = get_threads_util( $orig_master_handler->{dbh},
         $orig_master_handler->{connection_id} );
      }
```

```
 ##Terminating all threads
      printcurrent_time_us() . " Killing all application threads..\n";
      $orig_master_handler->kill_threads(@threads)if ( $#threads >= 0 );
      printcurrent_time_us() . " done.\n";
     $orig_master_handler->enable_log_bin_local();
     $orig_master_handler->disconnect();

      ## Afterfinishing the script, MHA executes FLUSH TABLES WITH READ LOCK
     $exit_code = 0;
    };
    if ($@) {
      warn"Got Error: $@\n";
      exit$exit_code;
    }
    exit$exit_code;
  }
  elsif ($command eq "start" ) {
    ##Activating master ip on the new master
    # 1. Createapp user with write privileges
    # 2. Movingbackup script if needed
    # 3.Register new master's ip to the catalog database
```

```
# We don't return error even though activatingupdatable accounts/ip failed so that we don't interrupt slaves' recovery.
# If exit code is 0 or 10, MHA does not abort
    my$exit_code = 10;
    eval {
      my$new_master_handler = new MHA::DBHelper();

      # args:hostname, port, user, password, raise_error_or_not
     $new_master_handler->connect( $new_master_ip, $new_master_port,
       $new_master_user, $new_master_password, 1 );

      ## Setread_only=0 on the new master
     $new_master_handler->disable_log_bin_local();
      printcurrent_time_us() . " Set read_only=0 on the new master.\n";
     $new_master_handler->disable_read_only();

      ##Creating an app user on the new master
      #printcurrent_time_us() . " Creating app user on the new master..\n";
      printcurrent_time_us() . "Add vip $vip on $if..\n";
     #create_app_user($new_master_handler);
     &add_vip();
     $new_master_handler->enable_log_bin_local();
     $new_master_handler->disconnect();
```



```
## Updatemaster ip on the catalog database, etc
     $exit_code = 0;
    };
    if ($@) {
      warn"Got Error: $@\n";
      exit$exit_code;
    }
    exit$exit_code;
  }
  elsif ($command eq "status" ) {

    # donothing
    exit 0;
  }
  else {
   &usage();
    exit 1;
  }
}
```

```
sub usage {
  print
"Usage:master_ip_online_change --command=start|stop|status --orig_master_host=host--orig_master_ip=ip --orig_master_port=port --new_master_host=host--new_master_ip=ip --new_master_port=port\n";
  die;
}
```

完事儿过后，就可以给两个脚本增加权限：

```
chmod +x master_ip_failover

chmod +x master_ip_online_change
```

接着安装需要的软件包：yum -y install perl-Time-HiRes

执行SSH检测命令：/usr/local/bin/masterha_check_ssh --conf=/etc/mha/mha.conf

如果检测结果全部显示为OK，那么就代表你安装完毕了

然后检测主从架构：/usr/local/bin/masterha_check_repl --conf=/etc/mha/mha.conf

如果检测结果全部正常，那么就代表没问题了

好，今天我们就讲解到这里，下次继续讲解

End