

# 一、背景

在我之前的博客 [git学习——> Gitlab如何进行备份恢复与迁移?](http://blog.csdn.net/ouyang_peng/article/details/77070977) (地址: [http://blog.csdn.net/ouyang\\_peng/article/details/77070977](http://blog.csdn.net/ouyang_peng/article/details/77070977)) 里面已经写清楚了如何使用Gitlab自动备份功能。

```
root@ubuntu4146:/data/gitlabData/backups# ll
total 14754836
drwx----- 3 git root      4096  8月 17 14:31 ./
drwx----- 5 git root      4096  8月 11 09:11 ../
-rwxrwxrwx 1 root root 1014190080  8月 10 17:43 1502357536_2017_08_10_9.4.3_gitlab_backup.tar*
-rw----- 1 git git 1014323200  8月 11 09:12 1502413978_2017_08_11_9.4.3_gitlab_backup.tar
-rw----- 1 git git 1014323200  8月 11 09:14 1502414088_2017_08_11_9.4.3_gitlab_backup.tar
-rw----- 1 git git 1182822400  8月 11 18:24 1502447053_2017_08_11_9.4.3_gitlab_backup.tar
-rw----- 1 git git 1183027200  8月 12 15:32 1502523125_2017_08_12_9.4.3_gitlab_backup.tar
-rw----- 1 git git 1183088640  8月 12 18:11 1502532676_2017_08_12_9.4.3_gitlab_backup.tar
-rw----- 1 git git 1183406080  8月 14 10:20 1502677230_2017_08_14_9.4.3_gitlab_backup.tar
-rw----- 1 git git 1086392320  8月 14 19:17 1502709438_2017_08_14_9.4.3_gitlab_backup.tar
-rw----- 1 git git 1087180800  8月 15 10:30 1502764235_2017_08_15_9.4.3_gitlab_backup.tar
-rw----- 1 git git 1253898240  8月 15 10:35 1502764533_2017_08_15_9.4.3_gitlab_backup.tar
-rw----- 1 git git 1252986880  8月 15 10:39 1502764773_2017_08_15_9.4.3_gitlab_backup.tar
-rw----- 1 git git 1251266560  8月 16 02:00 1502820036_2017_08_16_9.4.3_gitlab_backup.tar
-rw----- 1 git git 1401958400  8月 17 02:00 1502906429_2017_08_17_9.4.3_gitlab_backup.tar
-rwxr-xr-x 1 root root      33  8月 15 10:28 auto_backup.sh*
-rwxrwxrwx 1 root root     1329  8月 17 14:31 auto_backup_to_remote.sh*
drwxr-xr-x 2 root root      4096  8月 17 14:30 log/
-rw-r--r-- 1 root root      153  8月 14 10:22 README.txt
root@ubuntu4146:/data/gitlabData/backups#
```

定时脚本正常执行后，每天凌晨2点自动备份Gitlab

[http://blog.csdn.net/ouyang\\_peng](http://blog.csdn.net/ouyang_peng)

但是之前的备份功能只是备份到Gitlab服务运行的那台服务器上，如果哪一天那台服务器的磁盘损坏了的话，数据无法取出，那么对于公司来说是一匹无法想象的损失，因为

代码是公司的重要资产，需要以防万一。  
代码是公司的重要资产，需要以防万一。  
代码是公司的重要资产，需要以防万一。

因此我们得做好代码的备份工作，因此除了每天在Gitlab那台服务器上自动备份之外，还需要将每天的备份文件copy到另外一台文件备份服务器上，已达到双保险的要求。

## 二、服务器密钥配对，取消scp传输密码限制

远程手动备份数据费时费力且不及时。最好的方法就是通过脚本实现远程自动互备。但远程无论是通过SSH登陆，还是通过scp拷贝文件都需要输入密码。为了克服这个问题，首先需要实现不需要密码的SSH登陆，这样就可以使用 rsync, scp, rexec等命令来做的远程备份了。

前提：本地服务器：A， 远程服务器：B

## 2.1 生成密钥对

假设A， B两服务器，现在需要在A机上用root登陆B机，而不需要输入密码。那我们可按照下面的步骤来做：

### 2.1.1 在本地服务器A上生成rsa证书

在本地服务器A上生成rsa证书，运行命令：

```
1  ssh-keygen -t rsa
2  1
```

完整运行如下所示：

```
1  root@ubuntu4146:/data/gitlabData/backups# ssh-keygen -t rsa
2  Generating public/private rsa key pair.
3  Enter file in which to save the key (/root/.ssh/id_rsa):
4  /root/.ssh/id_rsa already exists.
5  Overwrite (y/n)? y
6  Enter passphrase (empty for no passphrase):
7  Enter same passphrase again:
8  Your identification has been saved in /root/.ssh/id_rsa.
9  Your public key has been saved in /root/.ssh/id_rsa.pub.
10 The key fingerprint is:
11 50:75:d3:53:d7:d8:98:1f:e7:9f:43:19:31:0d:e1:c4 root@ubuntu4146
12 The key's randomart image is:
13 +--[ RSA 2048 ]-----+
14 |          ... oo+@* |
15 |          .    . +EoB |
```

```
16 |      .      .+=|
17 |      .      oo|
18 |      S      . o|
19 |            o.|
20 |            .|
21 |            |
22 |            |
23 +-----+
24 root@ubuntu4146:/data/gitlabData/backups#
25 1
26 2
27 3
28 4
29 5
30 6
31 7
32 8
33 9
34 10
35 11
36 12
37 13
38 14
39 15
40 16
41 17
42 18
43 19
44 20
45 21
46 22
47 23
48 24
49 25
```

```

root@ubuntu4146:/data/gitlabData/backups# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
/root/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
50:75:d3:53:d7:d8:98:1f:e7:9f:43:19:31:0d:e1:c4 root@ubuntu4146
The key's randomart image is:
+--[ RSA 2048 ]-----+
|      . . . oo+@* |
|      . . . +EoB |
|      . . . .+= |
|      . . . oo |
|      S . . o |
|      . . . o. |
|      . . . . |
|      . . . . |
+-----+
root@ubuntu4146:/data/gitlabData/backups# █

```

[http://blog.csdn.net/ouyang\\_peng](http://blog.csdn.net/ouyang_peng)

- 1、生成的过程中提示输入密钥对保存位置，直接回车，接受默认值就行了。
- 2、因为之前已经有/root/.ssh/id\_rsa 文件存在，因此提示你是否覆盖，输入y表示覆盖
- 3、接着会提示输入一个密码，直接回车，让它空着。当然，也可以输入一个密码。
- 4、接着输入确认密码，输入完之后，回车密钥对就生成完了。

在/root/.ssh下生成id\_rsa 和 id\_rsa.pub 两个文件，  
其中公共密钥保存在 /root/.ssh/id\_rsa.pub，私有密钥保存在/root/.ssh/id\_rsa。

```

root@ubuntu4146:~/ssh# pwd
/root/.ssh
root@ubuntu4146:~/ssh# ll
total 20
drwx----- 2 root root 4096 8月 17 14:53 ./
drwx----- 8 root root 4096 8月 17 14:31 ../
-rw----- 1 root root 1679 8月 17 14:45 id_rsa
-rw-r--r-- 1 root root 397 8月 17 14:45 id_rsa.pub
-rw-r--r-- 1 root root 888 8月 17 11:45 known_hosts
root@ubuntu4146:~/ssh# █

```

刚才生成的公钥私钥

[http://blog.csdn.net/ouyang\\_peng](http://blog.csdn.net/ouyang_peng)

## 2.1.2 在本地服务器A上cp生成rsa公钥证书

然后在/root/.ssh下复制备份一份id\_rsa.pub 命名为 id\_rsa.pub.A，以便拷贝到远程服务器B。

## 执行cp命令复制

```
1 cp id_rsa.pub id_rsa.pub.A
2 1
```

```
root@ubuntu4146:~/ssh# ll
total 20
drwx----- 2 root root 4096 8月 17 14:53 ./
drwx----- 8 root root 4096 8月 17 14:31 ../
-rw----- 1 root root 1679 8月 17 14:45 id_rsa
-rw-r--r-- 1 root root 397 8月 17 14:45 id_rsa.pub
-rw-r--r-- 1 root root 888 8月 17 11:45 known_hosts
root@ubuntu4146:~/ssh# cp id_rsa.pub id_rsa.pub.A
root@ubuntu4146:~/ssh# ll
total 24
drwx----- 2 root root 4096 8月 17 14:54 ./
drwx----- 8 root root 4096 8月 17 14:31 ../
-rw----- 1 root root 1679 8月 17 14:45 id_rsa
-rw-r--r-- 1 root root 397 8月 17 14:45 id_rsa.pub
-rw-r--r-- 1 root root 397 8月 17 14:54 id_rsa.pub.A
-rw-r--r-- 1 root root 888 8月 17 11:45 known_hosts
root@ubuntu4146:~/ssh#
```

复制 id\_rsa.pub 为 id\_rsa.pub.A

[http://blog.csdn.net/ouyang\\_peng](http://blog.csdn.net/ouyang_peng)

## 2.2 cp生成rsa公钥证书到远程服务器B

使用scp命令进行远程复制，将A机生成的id\_rsa.pub.A拷贝到远程服务器B的/root/.ssh目录下

```
1 root@ubuntu4146:~/ssh# scp /root/.ssh/id_rsa.pub.A root@远程服务器
ip:/root/.ssh/
2 root@远程服务器ip's password:
3 id_rsa.pub.A
4 1
5 2
6 3
```

```
root@ubuntu4146:~/ssh# scp /root/.ssh/id_rsa.pub.A root@172.28.1.1:/root/.ssh/
root@172.28.1.1's password:
id_rsa.pub.A
root@ubuntu4146:~/ssh#
```

[http://blog.csdn.net/ouyang\\_peng](http://blog.csdn.net/ouyang_peng)

这里使用scp命令需要输入密码，当我们把下面的第三步执行完毕之后，以后本地服务器A使用scp命令复制文件到远程服务器B的话，就不需要再次输入密码。

## 三、密钥配对

### 3.1 创建authorized\_keys文件

当第二步将服务器A上的id\_rsa.pub.A 文件copy到了服务器B的目录/root/.ssh下之后截图如下：

```
[root@xtgl207940 .ssh]# pwd
/root/.ssh
[root@xtgl207940 .ssh]# ll
总用量 20
drwxr-xr-x. 2 root root 4096 8月 17 09:09 backup
-rw-r--r--. 1 root root 3243 7月 27 16:00 id_rsa
-rw-r--r--. 1 root root 758 7月 27 16:00 id_rsa.pub
-rw-r--r--. 1 root root 397 8月 17 14:57 id_rsa.pub.A
-rw-r--r--. 1 root root 1092 8月 11 11:22 known_hosts
[root@xtgl207940 .ssh]#
```

刚刚从服务器A scp过来的id\_rsa.pub.A文件

[http://blog.csdn.net/ouyang\\_peng](http://blog.csdn.net/ouyang_peng)

现在我们在 B 的/root/.ssh下创建authorized\_keys文件，使用如下命令

```
1 touch authorized_keys
2
```

```
[root@xtgl207940 .ssh]# pwd
/root/.ssh
[root@xtgl207940 .ssh]# ll
总用量 20
drwxr-xr-x. 2 root root 4096 8月 17 09:09 backup
-rw-r--r--. 1 root root 3243 7月 27 16:00 id_rsa
-rw-r--r--. 1 root root 758 7月 27 16:00 id_rsa.pub
-rw-r--r--. 1 root root 397 8月 17 14:57 id_rsa.pub.A
-rw-r--r--. 1 root root 1092 8月 11 11:22 known_hosts
[root@xtgl207940 .ssh]# touch authorized_keys
[root@xtgl207940 .ssh]# ll
总用量 20
-rw-r--r--. 1 root root 0 8月 17 15:05 authorized_keys
drwxr-xr-x. 2 root root 4096 8月 17 09:09 backup
-rw-r--r--. 1 root root 3243 7月 27 16:00 id_rsa
-rw-r--r--. 1 root root 758 7月 27 16:00 id_rsa.pub
-rw-r--r--. 1 root root 397 8月 17 14:57 id_rsa.pub.A
-rw-r--r--. 1 root root 1092 8月 11 11:22 known_hosts
[root@xtgl207940 .ssh]#
```

创建authorized\_keys文件

[http://blog.csdn.net/ouyang\\_peng](http://blog.csdn.net/ouyang_peng)

### 3.2 将id\_rsa.pub.A文件内容追加到authorized\_keys 文件中

通过 cat 命令 把id\_rsa.pub.A 追写到 authorized\_keys 文件中，命令依次如下：

```
1 cat id_rsa.pub.A >> authorized_keys
2 1
```

```
[root@xtgl207940 .ssh]# cat id_rsa.pub.A >> authorized_keys
[root@xtgl207940 .ssh]# cat id_rsa.pub.A >> authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQgAACoE1D9HwM6NEvpGCRZKj7bBa2JmVJ24gsUBIfptmUAskxyVhK1hWkMVCsUuyKLG5o7v3s7nVooq3o7cm07o2oVWEVRdeudKksrtcbzhGkVgYzypAQDTp/V1PX2q0I2dg3NwXm2/G1F
tst1pE0m1431vXCHMD0ZB717qXW6Zoonj1b7C4s0x07J52VgAShE33p34m1W6Dns7w3o7o7cm07o2oVWEVRdeudKksrtcbzhGkVgYzypAQDTp/V1PX2q0I2dg3NwXm2/G1F
dp 7e8e3b3b3e416
[root@xtgl207940 .ssh]#
```

### 3.3 修改authorized\_keys文件的权限

执行如下命令，修改authorized\_keys文件的权限

```
1 chmod 400 authorized_keys
2 1
```

```
[root@xtgl207940 .ssh]# chmod 400 authorized_keys
[root@xtgl207940 .ssh]# ll
总用量 24
-r----- 1 root root 397 8月 17 15:07 authorized_keys
drwxr-xr-x 2 root root 4096 8月 17 09:09 backup
-rw----- 1 root root 3243 7月 27 16:00 id_rsa
-rw-r--r-- 1 root root 758 7月 27 16:00 id_rsa.pub
-rw-r--r-- 1 root root 397 8月 17 14:57 id_rsa.pub.A
-rw-r--r-- 1 root root 1092 8月 11 11:22 known_hosts
[root@xtgl207940 .ssh]#
```

authorized\_keys文件的权限很重要，如果设置为777，那么登录的时候，还是需要提供密码的。

### 3.4 测试

测试服务器A使用scp命令复制文件到服务器B是否还需要密码

在服务A上，再次使用刚才的命令，发现已经可以不需要输入密码，如下所示：

```
root@ubuntu4146:~/.ssh# scp /root/.ssh/id_rsa.pub.A root@172.28.1.100:/root/.ssh/
root@172.28.1.100:~# s password:
id_rsa.pub.A
root@ubuntu4146:~/.ssh# scp /root/.ssh/id_rsa.pub.A root@172.28.1.100:/root/.ssh/
id_rsa.pub.A
root@ubuntu4146:~/.ssh#
```

第一次没有设置ssh证书授权认证的时候，是需要输入密码的

第二次设置好授权之后，不需要输入密码

设置之后，再次运行命令，不需要输入密码

100% 397 0.4KB/s 00:00

100% 397 0.4KB/s 00:00

[http://blog.csdn.net/nyang\\_peng](http://blog.csdn.net/nyang_peng)

## 四、创建Shell定时远程备份脚本

### 4.1 在本地服务器A上创建定时远程备份脚本

本地服务器A上创建定期备份脚本auto\_backup\_to\_remote.sh，脚本内容如下

```
1  #!/bin/bash
2
3  # gitlab 机房备份路径
4  LocalBackDir=/data/gitlabData/backups
5
6  # 远程备份服务器 gitlab备份文件存放路径
7  RemoteBackDir=/root/gitlabDataBackup
8
9  # 远程备份服务器 登录账户
10 RemoteUser=root
11
12 # 远程备份服务器 IP地址
13 RemoteIP=（你的远程服务器地址）请自己修改
14
15 #当前系统日期
16 DATE=`date +"%Y-%m-%d"`
17
18 #Log存放路径
19 LogFile=$LocalBackDir/log/$DATE.log
20
21 # 查找 本地备份目录下 时间为60分钟之内的，并且后缀为.tar的gitlab备份文件
22 BACKUPFILE_SEND_TO_REMOTE=$(find /data/gitlabData/backups -type f -mmin -60 -
name '*.tar*')
23
24 #新建日志文件
```



```

25 touch $LogFile
26
27 #追加日志到日志文件
28 echo "Gitlab auto backup to remote server, start at $(date +"%Y-%m-%d
%H:%M:%S")" >> $LogFile
29 echo "-----
---" >> $LogFile
30
31 # 输出日志，打印出每次scp的文件名
32 echo "-----The file to scp to remote server is:
$BACKUPFILE_SEND_TO_REMOTE-----" >> $LogFile
33
34
35 #备份到远程服务器
36 scp $BACKUPFILE_SEND_TO_REMOTE $RemoteUser@$RemoteIP:$RemoteBackDir
37
38 #追加日志到日志文件
39 echo "-----
---" >> $LogFile
40

```

```

#!/bin/bash
# gitlab 机房备份路径
LocalBackDir=/data/gitlabData/backups
# 远程备份服务器 gitlab备份文件存放路径
RemoteBackDir=/root/gitlabDataBackup
# 远程备份服务器 登录账户
RemoteUser=root
# 远程备份服务器 IP地址
RemoteIP=172.28.
#当前系统日期
DATE=`date +"%Y-%m-%d"`
#Log存放路径
LogFile=$LocalBackDir/log/$DATE.log
# 查找 本地备份目录下 时间为60分钟之内的，并且后缀为.tar的gitlab备份文件
BACKUPFILE_SEND_TO_REMOTE=$(find /data/gitlabData/backups -type f -mmin -1000 -name '*.tar*')
#新建日志文件
touch $LogFile
#追加日志到日志文件
echo "Gitlab auto backup to remote server, start at $(date +"%Y-%m-%d %H:%M:%S")" >> $LogFile
echo "-----" >> $LogFile
# 输出日志，打印出每次scp的文件名
echo "-----The file to scp to remote server is: $BACKUPFILE_SEND_TO_REMOTE-----" >> $LogFile
#备份到远程服务器
scp $BACKUPFILE_SEND_TO_REMOTE $RemoteUser@$RemoteIP:$RemoteBackDir
#追加日志到日志文件
echo "-----" >> $LogFile
"auto_backup_to_remote.sh" 41L, 1335C

```

[http://blog.csdn.net/ouyang\\_peng](http://blog.csdn.net/ouyang_peng)

因为到时候，我们会将该定时远程备份脚本auto\_backup\_to\_remote.sh执行的时间，放到Gitlab自动备份脚本auto\_backup.sh之后的一小时之内，因此我们只需要每次执行远程备份脚本auto\_backup\_to\_remote.sh的时候，只需要cp一个小时之内生成的新的Gitlab备份文件。

## 4.2 修改定时远程备份脚本

# 本auto\_backup\_to\_remote.sh的权限

要执行脚本文件，需要修改定时远程备份脚本auto\_backup\_to\_remote.sh的权限

```
1 chmod 777 auto_backup_to_remote.sh
2 1
```

## 4.3 手动执行脚本

现在为了验证脚本是否可以正常运行，我们需要手动执行脚本。

```
root@ubuntu4146:/data/gitlabData/backups# ll
total 14754836
drwx----- 3 git root 4096 8月 17 15:27 ./
drwx----- 5 git root 4096 8月 11 09:11 ../
-rwxrwxrwx 1 root root 1014190080 8月 10 17:43 1502357536 2017 08 10 9.4.3 gitlab backup.tar*
-rw----- 1 git git 1014323200 8月 11 09:12 1502413978 2017 08 11 9.4.3 gitlab backup.tar
-rw----- 1 git git 1014323200 8月 11 09:14 1502414088 2017 08 11 9.4.3 gitlab backup.tar
-rw----- 1 git git 1182822400 8月 11 18:24 1502447053 2017 08 11 9.4.3 gitlab backup.tar
-rw----- 1 git git 1183027200 8月 12 15:32 1502523125 2017 08 12 9.4.3 gitlab backup.tar
-rw----- 1 git git 1183088640 8月 12 18:11 1502532676 2017 08 12 9.4.3 gitlab backup.tar
-rw----- 1 git git 1183408000 8月 14 10:20 1502677230 2017 08 14 9.4.3 gitlab backup.tar
-rw----- 1 git git 1086392320 8月 14 19:17 1502709438 2017 08 14 9.4.3 gitlab backup.tar
-rw----- 1 git git 1087180800 8月 15 10:30 1502764235 2017 08 15 9.4.3 gitlab backup.tar
-rw----- 1 git git 1253898240 8月 15 10:35 1502764533 2017 08 15 9.4.3 gitlab backup.tar
-rw----- 1 git git 1252986880 8月 15 10:39 1502764773 2017 08 15 9.4.3 gitlab backup.tar
-rw----- 1 git git 1251266560 8月 16 02:00 1502820036 2017 08 16 9.4.3 gitlab backup.tar
-rw----- 1 git git 1401958400 8月 17 02:00 1502906429 2017 08 17 9.4.3 gitlab backup.tar
-rwxr-xr-x 1 root root 33 8月 15 10:28 auto_backup.sh*
-rwxrwxrwx 1 root root 1335 8月 17 15:18 auto_backup_to_remote.sh*
drwxr-xr-x 2 root root 4096 8月 17 15:21 log/
-rw-r--r-- 1 root root 153 8月 14 10:22 README.txt
root@ubuntu4146:/data/gitlabData/backups# date
2017年 08月 17日 星期四 15:27:16 CST
root@ubuntu4146:/data/gitlabData/backups#
```

因为最后一次备份时间是凌晨2点，而现在时间是下午3点，因此我们需要修改脚本的find查询条件

[http://blog.csdn.net/ouyang\\_peng](http://blog.csdn.net/ouyang_peng)

如上图所示，因为最后一次Gitlab备份时间为凌晨2点，而现在已经到了下午3点半左右，因此我们需要修改脚本的查询条件。

将查询条件从 本地备份目录下 时间为**60分钟之内**的，并且后缀为.tar的gitlab备份文件

```
1 # 查找 本地备份目录下 时间为60分钟之内的，并且后缀为.tar的gitlab备份文件
2 BACKUPFILE_SEND_TO_REMOTE=$(find /data/gitlabData/backups -type f -mmin -1000
  -name '*.tar*')
3 1
4 2
```

修改为 本地备份目录下 时间为**1000分钟**之内的，并且后缀为.tar的gitlab备份文件

```
1 # 查找 本地备份目录下 时间为1000分钟之内的，并且后缀为.tar的gitlab备份文件
2 BACKUPFILE_SEND_TO_REMOTE=$(find /data/gitlabData/backups -type f -mmin -1000
  -name '*.tar*')
3 1
4 2
```

先在终端执行find命令，看是否能够正常查找出我们要scp到远程服务器的Gitlab备份文件

```
1 root@ubuntu4146:/data/gitlabData/backups# find /data/gitlabData/backups -type
  f -mmin -1000 -name '*.tar*'
2 /data/gitlabData/backups/1502906429_2017_08_17_9.4.3_gitlab_backup.tar
3 1
4 2
5 3
```

```
root@ubuntu4146:/data/gitlabData/backups# ll
total 14754836
drwx----- 3 git root 4096 8月 17 15:27 ./
drwx----- 5 git root 4096 8月 11 09:11 ../
-rwxrwxrwx 1 root root 1014190080 8月 10 17:43 1502357536_2017_08_10_9.4.3_gitlab_backup.tar*
-rw----- 1 git git 1014323200 8月 11 09:12 1502413978_2017_08_11_9.4.3_gitlab_backup.tar
-rw----- 1 git git 1014323200 8月 11 09:14 1502414088_2017_08_11_9.4.3_gitlab_backup.tar
-rw----- 1 git git 1182822400 8月 11 18:24 1502447053_2017_08_11_9.4.3_gitlab_backup.tar
-rw----- 1 git git 1183027200 8月 12 15:32 1502523125_2017_08_12_9.4.3_gitlab_backup.tar
-rw----- 1 git git 1183080640 8月 12 18:11 1502532676_2017_08_12_9.4.3_gitlab_backup.tar
-rw----- 1 git git 1183406080 8月 14 10:20 1502677230_2017_08_14_9.4.3_gitlab_backup.tar
-rw----- 1 git git 1086392320 8月 14 19:17 1502709438_2017_08_14_9.4.3_gitlab_backup.tar
-rw----- 1 git git 1087180800 8月 15 10:30 1502764235_2017_08_15_9.4.3_gitlab_backup.tar
-rw----- 1 git git 1253898240 8月 15 10:35 1502764533_2017_08_15_9.4.3_gitlab_backup.tar
-rw----- 1 git git 1252986880 8月 15 10:39 1502764773_2017_08_15_9.4.3_gitlab_backup.tar
-rw----- 1 git git 1251266560 8月 16 02:00 1502820036_2017_08_16_9.4.3_gitlab_backup.tar
-rw----- 1 git git 1401958400 8月 17 02:00 1502906429_2017_08_17_9.4.3_gitlab_backup.tar
-rwxr-xr-x 1 root root 33 8月 15 10:28 auto_backup.sh*
-rwxrwxrwx 1 root root 1335 8月 17 15:18 auto_backup_to_remote.sh*
drwxr-xr-x 2 root root 4096 8月 17 15:21 log/
-rw-r--r-- 1 root root 153 8月 14 10:22 README.txt
root@ubuntu4146:/data/gitlabData/backups# date
2017年 08月 17日 星期四 15:27:16 CST
root@ubuntu4146:/data/gitlabData/backups# vi auto_backup_to_remote.sh
root@ubuntu4146:/data/gitlabData/backups# find /data/gitlabData/backups -type f -mmin -1000 -name '*.tar*'
/data/gitlabData/backups/1502906429_2017_08_17_9.4.3_gitlab_backup.tar
root@ubuntu4146:/data/gitlabData/backups#
```

手动执行find命令，列出本次将要scp的备份文件

[http://blog.csdn.net/ouyang\\_peng](http://blog.csdn.net/ouyang_peng)

将定时远程备份脚本auto\_backup\_to\_remote.sh修改完毕之后，我们试着手动执行该脚本，看是否能够正常运行。

执行命令

```

1 root@ubuntu4146:/data/gitlabData/backups# ./auto_backup_to_remote.sh
2 1

```

```

root@ubuntu4146:/data/gitlabData/backups# ./auto_backup_to_remote.sh
1502906429_2017_08_17_9.4.3_gitlab_backup.tar

```

5% 0, 69MB, 12.4MB/s, 01:42, ETA

执行该脚本文件，在2分钟之内就将最后一次的Gitlab备份文件scp到了远程服务器B

```

1 root@ubuntu4146:/data/gitlabData/backups# ./auto_backup_to_remote.sh
2 1502906429_2017_08_17_9.4.3_gitlab_backup.tar

100% 1337MB 11.2MB/s 01:59
3 1
4 2

```

```

root@ubuntu4146:/data/gitlabData/backups# ./auto_backup_to_remote.sh
1502906429_2017_08_17_9.4.3_gitlab_backup.tar

```

100% 1337MB, 11.2MB/s, 01:59, peng

我们切换到远程服务器B，查看刚才从服务器A 通过scp命令复制过来的Gitlab备份文件，如下所示：

```

[root@xtgl207940 .ssh]# cd /root/gitlabDataBackup/
[root@xtgl207940 gitlabDataBackup]# ll
总用量 1369100
-rw-----. 1 root root 1401958400 8月 17 15:36
[root@xtgl207940 gitlabDataBackup]#

```

1502906429\_2017\_08\_17\_9.4.3\_gitlab\_backup.tar

从服务器A scp过来的备

[http://blog.csdn.net/ouyang\\_peng](http://blog.csdn.net/ouyang_peng)

```

1 [root@xtgl207940 .ssh]# cd /root/gitlabDataBackup/
2 [root@xtgl207940 gitlabDataBackup]# ll
3 总用量 1369100
4 -rw-----. 1 root root 1401958400 8月 17 15:36
5 1502906429_2017_08_17_9.4.3_gitlab_backup.tar
6 [root@xtgl207940 gitlabDataBackup]#
7 1
8 2
9 3
10 4
11 5
12 6

```

至此，从服务器A复制备份文件到服务器B的脚本正常运行，但是如果每次都这样来手动触发脚本的话，太麻烦，因此我们需要定时执行脚本。

## 五、定时执行脚本

### 5.1 编辑/etc/crontab 文件

即vi /etc/crontab，然后添加相应的任务。

```
1 #编辑 /etc/crontab
2 vi /etc/crontab
3 1
4 2
```

可以看到，这里是我们之前写好的定期凌晨2点执行Gitlab本机备份的定时任务

```
1 # /etc/crontab: system-wide crontab
2 # Unlike any other crontab you don't have to run the `crontab'
3 # command to install the new version when you edit this file
4 # and files in /etc/cron.d. These files also have username fields,
5 # that none of the other crontabs do.
6
7 SHELL=/bin/sh
8 PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
9
10 # m h dom mon dow user  command
11 17 * * * * root    cd / && run-parts --report /etc/cron.hourly
12 25 6 * * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --
report /etc/cron.daily )
13 47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-parts --
report /etc/cron.weekly )
14 52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --
report /etc/cron.monthly )
15 #
16
17 # edited by ouyang 2017-8-11 添加定时任务，每天凌晨两点，执行gitlab备份
18 0 2 * * * root    /opt/gitlab/bin/gitlab-rake gitlab:backup:create
CRON=1
```

```

19
20 #也可以按照如下所示的方法，定时执行 auto_backup.sh脚本，脚本内容就填写：
    /opt/gitlab/bin/gitlab-rake gitlab:backup:create CRON=1
21 #0 2 * * * root /data/gitlabData/backups/auto_backup.sh -D 1
22 1
23 2
24 3
25 4
26 5
27 6
28 7
29 8
30 9
31 10
32 11
33 12
34 13
35 14
36 15
37 16
38 17
39 18
40 19
41 20
42 21
43 22

```

```

root@ubuntu4146:/data/gitlabData/backups# vi /etc/crontab

/etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6 * * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
# edited by ouyang 2017-8-11 添加定时任务，每天凌晨两点，执行gitlab备份
0 2 * * * root    /opt/gitlab/bin/gitlab-rake gitlab:backup:create CRON=1

#也可以按照如下所示的方法，定时执行 auto_backup.sh脚本，脚本内容就填写： /opt/gitlab/bin/gitlab-rake gitlab:backup:create CRON=1
#0 2 * * * root    /data/gitlabData/backups/auto_backup.sh -D 1

```

[http://blog.csdn.net/ouyang\\_peng](http://blog.csdn.net/ouyang_peng)

现在我们在上面的定时任务后面，再添加一个执行复制刚备份好的Gitlab备份文件到服务器B的脚本任务。如下所示：

```

1 # edited by ouyang 2017-8-17 添加定时任务，每天凌晨三点，执行gitlab备份到远程服务

```

器

```
2 0 3 * * * root /data/gitlabData/backups/auto_backup_to_remote.sh
3 1
4 2
```

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6 * * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
# edited by ouyang 2017-8-11 添加定时任务，每天凌晨两点，执行gitlab备份
0 2 * * * root    /opt/gitlab/bin/gitlab-rake gitlab:backup:create CRON=1

#也可以按照如下所示的方法，定时执行 auto_backup.sh脚本，脚本内容就填写： /opt/gitlab/bin/gitlab-rake gitlab:backup:create CRON=1
#0 2 * * * root    /data/gitlabData/backups/auto_backup.sh -D 1

# edited by ouyang 2017-8-17 添加定时任务，每天凌晨三点，执行gitlab备份到远程服务器
0 3 * * * root    /data/gitlabData/backups/auto_backup_to_remote.sh
```

[http://blog.csdn.net/ouyang\\_peng](http://blog.csdn.net/ouyang_peng)

## 5.2 重启cron服务

编写完 /etc/crontab 文件之后，需要重新启动cron服务

```
1 #重新加载cron配置文件
2 sudo /usr/sbin/service cron reload
3 #重启cron服务
4 sudo /usr/sbin/service cron restart
5 1
6 2
7 3
8 4
```

实际运行如下：

```
1 root@ubuntu4146:/data/gitlabData/backups# sudo /usr/sbin/service cron reload
2 root@ubuntu4146:/data/gitlabData/backups# sudo /usr/sbin/service cron restart
3 cron stop/waiting
4 cron start/running, process 47631
5 1
6 2
```

```
7 3
8 4
9 5
```

```
root@ubuntu4146:/data/gitlabData/backups# vi /etc/crontab
root@ubuntu4146:/data/gitlabData/backups# sudo /usr/sbin/service cron reload
root@ubuntu4146:/data/gitlabData/backups# sudo /usr/sbin/service cron reload
root@ubuntu4146:/data/gitlabData/backups# sudo /usr/sbin/service cron restart
cron stop/waiting
cron start/running, process 47631
```

[http://blog.csdn.net/ouyang\\_peng](http://blog.csdn.net/ouyang_peng)

## 5.3 测试自动化脚本

为了能够测试，该脚本是否能够在指定时间的时候，真的能够自动执行，我们将时间修改为15:55分。修改如下：

```
1 # edited by ouyang 2017-8-17 添加定时任务，每天凌晨三点，执行gitlab备份到远程服务
   器
2 55 15 * * * root /data/gitlabData/backups/auto_backup_to_remote.sh
3 1
4 2
```

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6 * * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
# edited by ouyang 2017-8-11 添加定时任务，每天凌晨两点，执行gitlab备份
0 2 * * * root    /opt/gitlab/bin/gitlab-rake gitlab:backup:create CRON=1
#也可以按照如下所示的方法，定时执行 auto_backup.sh脚本，脚本内容就填写： /opt/gitlab/bin/gitlab-rake gitlab:backup:create CRON=1
#0 2 * * * root    /data/gitlabData/backups/auto_backup.sh -D 1

# edited by ouyang 2017-8-17 添加定时任务，每天凌晨三点，执行gitlab备份到远程服务器
55 15 * * * root    /data/gitlabData/backups/auto_backup_to_remote.sh
```

测试的时候，修改执行时间为15: 55

[http://blog.csdn.net/ouyang\\_peng](http://blog.csdn.net/ouyang_peng)

然后重启cron服务

```
1 #重新加载cron配置文件
2 sudo /usr/sbin/service cron reload
3 #重启cron服务
```



```
4 sudo /usr/sbin/service cron restart
5 1
6 2
7 3
8 4
```

现在时间是15:58分，我们去查看生成的log文件，可以看到在15:55分的时候，脚本正常定时执行了

```

1 Gitlab auto backup to remote server, start at 2017-08-17 15:55:01
2 -----
3 -----The file to scp to remote server is:
4 /data/gitlabData/backups/1502906429_2017_08_17_9.4.3_gitlab_backup.tar-----
5 -----
6 Gitlab auto backup to remote server, end at 2017-08-17 15:57:00
7 1
8 2
9 3
10 4
11 5
12 6

```

```

Gitlab auto backup to remote server, start at 2017-08-17 15:34:59
-----The file to scp to remote server is: /data/gitlabData/backups/1502906429_2017_08_17_9.4.3_gitlab_backup.tar-----
Gitlab auto backup to remote server, end at 2017-08-17 15:36:59
Gitlab auto backup to remote server, start at 2017-08-17 15:55:01
-----The file to scp to remote server is: /data/gitlabData/backups/1502906429_2017_08_17_9.4.3_gitlab_backup.tar-----
Gitlab auto backup to remote server, end at 2017-08-17 15:57:00

```

定时任务 在 15: 55分正常执行

"log/2017-08-17.log" 10L, 888C

[http://blog.csdn.net/ouyang\\_peng](http://blog.csdn.net/ouyang_peng)

切换到远程服务器B，查看从服务器A copy过来的Gitlab备份文件

```

[root@xtgl207940 .ssh]# cd /root/gitlabDataBackup/
[root@xtgl207940 gitlabDataBackup]# ll
总用量 1369100
-rw-r----- 1 root root 1401958400 8月 17 15:36 1502906429_2017_08_17_9.4.3_gitlab_backup.tar
[root@xtgl207940 gitlabDataBackup]# ll
总用量 1369100
-rw-r----- 1 root root 1401958400 8月 17 15:55 1502906429_2017_08_17_9.4.3_gitlab_backup.tar
[root@xtgl207940 gitlabDataBackup]#

```

15:55 执行定时任务后

[http://blog.csdn.net/ouyang\\_peng](http://blog.csdn.net/ouyang_peng)

通过测试，可以发现定时任务也正常执行了，因此我们可以将时间改为凌晨3点来复制Gitlab备份文件到远程服务器B。

```

1 # edited by ouyang 2017-8-17 添加定时任务，每天凌晨三点，执行gitlab备份到远程服务器
2 0 3 * * * root /data/gitlabData/backups/auto_backup_to_remote.sh
3 1
4 2

```

## 六、定时删除远程服务器上的备份文件

```

root@ubuntu4146:/data/gitlabData/backups# ll
total 14754836
drwx----- 3 git root 4096 8月 17 14:31 ./
drwx----- 5 git root 4096 8月 11 09:11 ../
-rwxrwxrwx 1 root root 1014190080 8月 10 17:43 1502357536_2017_08_10_9.4.3_gitlab_backup.tar*
-rw-r----- 1 git git 1014323200 8月 11 09:12 1502413978_2017_08_11_9.4.3_gitlab_backup.tar
-rw-r----- 1 git git 1014323200 8月 11 09:14 1502414088_2017_08_11_9.4.3_gitlab_backup.tar
-rw-r----- 1 git git 1182822400 8月 11 18:24 1502447053_2017_08_11_9.4.3_gitlab_backup.tar
-rw-r----- 1 git git 1183027200 8月 12 15:32 1502523125_2017_08_12_9.4.3_gitlab_backup.tar
-rw-r----- 1 git git 1183088640 8月 12 18:11 1502532676_2017_08_12_9.4.3_gitlab_backup.tar
-rw-r----- 1 git git 1183406080 8月 14 10:20 1502677230_2017_08_14_9.4.3_gitlab_backup.tar
-rw-r----- 1 git git 1086392320 8月 14 19:17 1502709438_2017_08_14_9.4.3_gitlab_backup.tar
-rw-r----- 1 git git 1087180800 8月 15 10:30 1502764235_2017_08_15_9.4.3_gitlab_backup.tar
-rw-r----- 1 git git 1253898240 8月 15 10:35 1502764533_2017_08_15_9.4.3_gitlab_backup.tar
-rw-r----- 1 git git 1252986880 8月 15 10:39 1502764773_2017_08_15_9.4.3_gitlab_backup.tar
-rw-r----- 1 git git 1251266560 8月 16 02:00 1502820036_2017_08_16_9.4.3_gitlab_backup.tar
-rw-r----- 1 git git 1401958400 8月 17 02:00 1502906429_2017_08_17_9.4.3_gitlab_backup.tar
-rwxr-xr-x 1 root root 33 8月 15 10:28 auto_backup.sh*
-rwxrwxrwx 1 root root 1329 8月 17 14:31 auto_backup_to_remote.sh*
drwxr-xr-x 2 root root 4096 8月 17 14:30 log/
-rw-r--r-- 1 root root 153 8月 14 10:22 README.txt
root@ubuntu4146:/data/gitlabData/backups#

```

定时脚本正常执行后，每天凌晨2点自动备份Gitlab

[http://blog.csdn.net/ouyang\\_peng](http://blog.csdn.net/ouyang_peng)

通过如上图所示，每个Gitlab备份文件都很大，都有1G左右的大小。因此每天备份一次，过不了多久的话，备份服务器B上的磁盘空间可能就会被Gitlab备份文件占用完。

因此我们需要定期清理备份文件，清理的时候我们可以自己定义，这里我们规定：

备份文件超过30天的都自动删除掉。

为了实现这个功能，我们需要在远程服务器B上编写脚本来清理过期的备份文件。

## 6.1 创建定期删除过期的备份文件的脚本

创建定期删除过期的备份文件的脚本**auto\_remove\_old\_backup.sh**

```
1 [root@xtgl207940 gitlabDataBackup]# touch auto_remove_old_backup.sh
2 1
3 2
```

```
[root@xtgl207940 gitlabDataBackup]# touch auto_remove_old_backup.sh
[root@xtgl207940 gitlabDataBackup]# ll
总用量 1369100
-rw-r--r-- 1 root root 1401958400 8月 17 15:56 1502906429_2017_08_17_9.4.3_gitlab_backup.tar
-rw-r--r-- 1 root root 0 8月 17 16:09 auto_remove_old_backup.sh
[root@xtgl207940 gitlabDataBackup]#
```

[http://blog.csdn.net/ouyang\\_peng](http://blog.csdn.net/ouyang_peng)

## 6.2 编写脚本auto\_remove\_old\_backup.sh

```
1 vi auto_remove_old_backup.sh
2 1
```

脚本内容如下：

```
1 #!/bin/bash
2
3 # 远程备份服务器 gitlab备份文件存放路径
4 GitlabBackDir=/root/gitlabDataBackup
5
6 # 查找远程备份路径下，超过30天 且文件后缀为.tar 的 Gitlab备份文件 然后删除
7 find $GitlabBackDir -type f -mtime +30 -name '*.tar*' -exec rm {} \;
8 1
9 2
```

```
10 3
11 4
12 5
13 6
14 7
15 8
```

## 6.3 手动执行auto\_remove\_old\_backup.sh脚本

修改auto\_remove\_old\_backup.sh脚本权限为777

```
1 chmod 777 auto_remove_old_backup.sh
2 1
```

```
[root@xtgl207940 gitlabDataBackup]# vi auto_remove_old_backup.sh
[root@xtgl207940 gitlabDataBackup]# ll
总用量 1369104
-rw-----. 1 root root 1401958400 8月 17 15:56 1502906429_2017_08_17_9.4.3_gitlab_backup.tar
-rwxrwxrwx. 1 root root 279 8月 17 16:24 auto_remove_old_backup.sh
[root@xtgl207940 gitlabDataBackup]#
```

[http://blog.csdn.net/ouyang\\_peng](http://blog.csdn.net/ouyang_peng)

如上图所示，目前备份服务器上，只有一个8月17日15:56分copy过来的Gitlab备份文件，为了能够测试我们的脚本是否正常运行，我们新建几个7月份的文件，如下所示：

使用touch命令，创建指定时间的 test.tar 文件

```
1 [root@xtgl207940 gitlabDataBackup]# touch -t 201707011230 test1.tar
2 [root@xtgl207940 gitlabDataBackup]# touch -t 201707021230 test2.tar
3 [root@xtgl207940 gitlabDataBackup]# touch -t 201707031230 test3.tar
4 [root@xtgl207940 gitlabDataBackup]# touch -t 201707041230 test4.tar
5 [root@xtgl207940 gitlabDataBackup]# ll
6 总用量 1369104
7 -rw-----. 1 root root 1401958400 8月 17 15:56
  1502906429_2017_08_17_9.4.3_gitlab_backup.tar
8 -rwxrwxrwx. 1 root root 279 8月 17 16:24 auto_remove_old_backup.sh
9 -rw-r--r--. 1 root root 0 7月 1 12:30 test1.tar
```

```

10 -rw-r--r--. 1 root root      0 7月  2 12:30 test2.tar
11 -rw-r--r--. 1 root root      0 7月  3 12:30 test3.tar
12 -rw-r--r--. 1 root root      0 7月  4 12:30 test4.tar
13 [root@xtgl207940 gitlabDataBackup]#
14 1
15 2
16 3
17 4
18 5
19 6
20 7
21 8
22 9
23 10
24 11
25 12
26 13
27 14

```

```

[root@xtgl207940 gitlabDataBackup]# touch -t 201707011230 test1.tar
[root@xtgl207940 gitlabDataBackup]# touch -t 201707021230 test2.tar
[root@xtgl207940 gitlabDataBackup]# touch -t 201707031230 test3.tar
[root@xtgl207940 gitlabDataBackup]# touch -t 201707041230 test4.tar
[root@xtgl207940 gitlabDataBackup]# ll
总用量 1369104
-rw-----. 1 root root 1401958400 8月 17 15:56 1502906429_2017_08_17_9.4.3_gitlab_backup.tar
-rwxrwxrwx. 1 root root      279 8月 17 16:24 auto_remove_old_backup.sh
-rw-r--r--. 1 root root      0 7月  1 12:30 test1.tar
-rw-r--r--. 1 root root      0 7月  2 12:30 test2.tar
-rw-r--r--. 1 root root      0 7月  3 12:30 test3.tar
-rw-r--r--. 1 root root      0 7月  4 12:30 test4.tar
[root@xtgl207940 gitlabDataBackup]#

```

[http://blog.csdn.net/ouyang\\_peng](http://blog.csdn.net/ouyang_peng)

这样我们就创建了4个7月1日到7月4日的的test1.tar、test2.tar、test3.tar、test4.tar

现在我们手动来执行我们的auto\_remove\_old\_backup.sh脚本

```

1 [root@xtgl207940 gitlabDataBackup]# ./auto_remove_old_backup.sh
2 [root@xtgl207940 gitlabDataBackup]# ll
3 总用量 1369104
4 -rw-----. 1 root root 1401958400 8月 17 15:56
  1502906429_2017_08_17_9.4.3_gitlab_backup.tar
5 -rwxrwxrwx. 1 root root      279 8月 17 16:24 auto_remove_old_backup.sh
6 [root@xtgl207940 gitlabDataBackup]#
7 1
8 2
9 3

```

```
10 4
11 5
12 6
13 7
```

```
[root@xtgl207940 gitlabDataBackup]# touch -t 201707011230 test1.tar
[root@xtgl207940 gitlabDataBackup]# touch -t 201707021230 test2.tar
[root@xtgl207940 gitlabDataBackup]# touch -t 201707031230 test3.tar
[root@xtgl207940 gitlabDataBackup]# touch -t 201707041230 test4.tar
[root@xtgl207940 gitlabDataBackup]# ll
总用量 1369104
-rw-----. 1 root root 1401958400 8月 17 15:56 1502906429_2017_08_17_9.4.3_gitlab_backup.tar
-rwxrwxrwx. 1 root root 279 8月 17 16:24 auto_remove_old_backup.sh
-rw-r--r--. 1 root root 0 7月 1 12:30 test1.tar
-rw-r--r--. 1 root root 0 7月 2 12:30 test2.tar
-rw-r--r--. 1 root root 0 7月 3 12:30 test3.tar
-rw-r--r--. 1 root root 0 7月 4 12:30 test4.tar
[root@xtgl207940 gitlabDataBackup]# ./auto_remove_old_backup.sh
[root@xtgl207940 gitlabDataBackup]# ll
总用量 1369104
-rw-----. 1 root root 1401958400 8月 17 15:56 1502906429_2017_08_17_9.4.3_gitlab_backup.tar
-rwxrwxrwx. 1 root root 279 8月 17 16:24 auto_remove_old_backup.sh
[root@xtgl207940 gitlabDataBackup]#
```

执行完脚本之后，30天之前的 \*.tar 文件都被删除了

[http://blog.csdn.net/ouyang\\_peng](http://blog.csdn.net/ouyang_peng)

对比执行auto\_remove\_old\_backup.sh脚本前后，我们发现超过30天的，并且以.tar后缀结尾的文件都被删除了，脚本正常。

## 七、定时执行删除脚本

### 7.1 编辑/etc/crontab 文件

即vi /etc/crontab，然后添加相应的任务。

```
1 #编辑 /etc/crontab
2 vi /etc/crontab
3 1
4 2
```

然后编写定时任务

```
1 SHELL=/bin/bash
2 PATH=/sbin:/bin:/usr/sbin:/usr/bin
3 MAILTO=root
4
5 # For details see man 4 crontabs
6
7 # Example of job definition:
```

```

8 # .----- minute (0 - 59)
9 # | .----- hour (0 - 23)
10 # | | .----- day of month (1 - 31)
11 # | | | .----- month (1 - 12) OR jan,feb,mar,apr ...
12 # | | | | .---- day of week (0 - 6) (Sunday=0 or 7) OR
    sun,mon,tue,wed,thu,fri,sat
13 # | | | | |
14 # * * * * * user-name  command to be executed
15
16
17 # edited by ouyang 2017-8-17 添加定时任务，每天凌晨4点，执行删除过期的Gitlab备份文件
18 0 4 * * * root /root/gitlabDataBackup/auto_remove_old_backup.sh
19 1
20 2
21 3
22 4
23 5
24 6
25 7
26 8
27 9
28 10
29 11
30 12
31 13
32 14
33 15
34 16
35 17
36 18

```

## 7.2 重启crond服务

写完 /etc/crontab 文件之后，需要重新启动cron服务，因为远程备份服务器是 Center OS 和之前的Gitlab服务器 Ubuntu 有点不一样，所以重启cron命令有所不同。

```

1 #重新加载cron配置文件
2 sudo service crond reload
3 #重启cron服务

```

```
4 sudo service crond restart
5 1
6 2
7 3
8 4
```

```
[root@xtgl207940 gitlabDataBackup]# vi /etc/crontab
[root@xtgl207940 gitlabDataBackup]# sudo service crond reload
Redirecting to /bin/systemctl reload crond.service
[root@xtgl207940 gitlabDataBackup]# sudo service crond restart
Redirecting to /bin/systemctl restart crond.service
[root@xtgl207940 gitlabDataBackup]#
```

[http://blog.csdn.net/ouyang\\_peng](http://blog.csdn.net/ouyang_peng)

## 7.3 测试定期删除任务

为了测试定期删除任务，现在时间是16:35，我们将脚本执行时间设置为16:40分，如下所示

```
1 # edited by ouyang 2017-8-17 添加定时任务，每天凌晨4点，执行删除过期的Gitlab备份文件
2 40 16 * * * root /root/gitlabDataBackup/auto_remove_old_backup.sh
3 1
4 2
5 3
```

然后重启cron服务，超过16点40分之后我们再查看过期的.tar文件是否被删除

```
[root@xtgl207940 gitlabDataBackup]# touch -t 201707011230 test1.tar
[root@xtgl207940 gitlabDataBackup]# touch -t 201707021230 test2.tar
[root@xtgl207940 gitlabDataBackup]# touch -t 201707031230 test3.tar
[root@xtgl207940 gitlabDataBackup]# touch -t 201707041230 test4.tar
[root@xtgl207940 gitlabDataBackup]# ll
总用量 1369104
-rw-----. 1 root root 1401958400 8月 17 15:56 1502906429_2017_08_17_9.4.3_gitlab_backup.tar
-rwxrwxrwx. 1 root root 279 8月 17 16:24 auto_remove_old_backup.sh
-rw-r--r--. 1 root root 0 7月 1 12:30 test1.tar
-rw-r--r--. 1 root root 0 7月 2 12:30 test2.tar
-rw-r--r--. 1 root root 0 7月 3 12:30 test3.tar
-rw-r--r--. 1 root root 0 7月 4 12:30 test4.tar
[root@xtgl207940 gitlabDataBackup]# vi /etc/
[root@xtgl207940 gitlabDataBackup]# sudo service crond reload
Redirecting to /bin/systemctl reload crond.service
[root@xtgl207940 gitlabDataBackup]# sudo service crond restart
Redirecting to /bin/systemctl restart crond.service
[root@xtgl207940 gitlabDataBackup]# ll
总用量 1369104
-rw-----. 1 root root 1401958400 8月 17 15:56 1502906429_2017_08_17_9.4.3_gitlab_backup.tar
-rwxrwxrwx. 1 root root 279 8月 17 16:24 auto_remove_old_backup.sh
[root@xtgl207940 gitlabDataBackup]#
```

修改定时任务时间，重启服务器后，在指定时间正常执行了删除超过30天的备份文件

[http://blog.csdn.net/ouyang\\_peng](http://blog.csdn.net/ouyang_peng)



```
1 [root@xtgl207940 gitlabDataBackup]# touch -t 201707011230 test1.tar
2 [root@xtgl207940 gitlabDataBackup]# touch -t 201707021230 test2.tar
3 [root@xtgl207940 gitlabDataBackup]# touch -t 201707031230 test3.tar
4 [root@xtgl207940 gitlabDataBackup]# touch -t 201707041230 test4.tar
5 [root@xtgl207940 gitlabDataBackup]# ll
6 总用量 1369104
7 -rw-----. 1 root root 1401958400 8月 17 15:56
  1502906429_2017_08_17_9.4.3_gitlab_backup.tar
8 -rwxrwxrwx. 1 root root 279 8月 17 16:24 auto_remove_old_backup.sh
9 -rw-r--r--. 1 root root 0 7月 1 12:30 test1.tar
10 -rw-r--r--. 1 root root 0 7月 2 12:30 test2.tar
11 -rw-r--r--. 1 root root 0 7月 3 12:30 test3.tar
12 -rw-r--r--. 1 root root 0 7月 4 12:30 test4.tar
13 [root@xtgl207940 gitlabDataBackup]# vi /etc/
14 [root@xtgl207940 gitlabDataBackup]# vi /etc/crontab
15 [root@xtgl207940 gitlabDataBackup]# sudo service crond reload
16 Redirecting to /bin/systemctl reload crond.service
17 [root@xtgl207940 gitlabDataBackup]# sudo service crond restart
18 Redirecting to /bin/systemctl restart crond.service
19 [root@xtgl207940 gitlabDataBackup]# ll
20 总用量 1369104
21 -rw-----. 1 root root 1401958400 8月 17 15:56
  1502906429_2017_08_17_9.4.3_gitlab_backup.tar
22 -rwxrwxrwx. 1 root root 279 8月 17 16:24 auto_remove_old_backup.sh
23 [root@xtgl207940 gitlabDataBackup]#
24 1
25 2
26 3
27 4
28 5
29 6
30 7
31 8
32 9
33 10
34 11
35 12
36 13
37 14
38 15
39 16
40 17
41 18
42 19
43 20
```

```
44 21
45 22
46 23
```

如上所示，我们修改定时执行任务的时间后，删除任务正常执行，因此我们时间修改回凌晨4点删除过期备份文件。

```
1 # edited by ouyang 2017-8-17 添加定时任务，每天凌晨4点，执行删除过期的Gitlab备份文件
2 0 4 * * * root /root/gitlabDataBackup/auto_remove_old_backup.sh
3 1
4 2
```

```
1 来源: http://blog.csdn.net/ouyang\_peng/article/details/77334215
2
```