案例一：备份MySQL数据库

一台MySQL服务器，跑着5个数据库，在没有做主从的情况下，需要对这5个库进行备份，具体要求如下：

1）每天备份一次，需要备份所有的库

2）把备份数据存放到/data/backup/下

3）备份文件名称格式示例：database\_name-2018-11-11.sql

4）需要对1天以前的所有sql文件压缩，格式为gzip

5）本地数据保留1周

6）需要把备份的数据同步到远程备份中心，假如本机可以直接通过rsync命令同步，同步目标地址为rsync://10.10.20.100/mysqlbak/

7）远程备份数据要求保留1个月

拿到需求后，我们不应该马上就动手，而是应该先分析需求点，准备好储备知识。如果你没有足够的储备知识，无论脚本语法掌握的多么熟练，这个脚本也是不能完成的。本脚本的核心知识点是如何备份所有的库。

知识点一：MySQL数据库备份

假设已知备份数据的用户为backup，密码为34KpmyzUq，要备份的5个数据库分别为db1-db5。备份数据库的命令为：

mysqldump -ubackup -p34KpmyzUq db1 > db1.sql

如果是远程的MySQL服务，还可以加上IP和Port，例如：

mysqldump -ubackup -p34KpmyzUq -h192.168.100.100 -P3308 db1 > db1.sql

若MySQL数据量很大，比如有几十GB，使用mysqldump备份工具会特别慢，推荐使用percona-xtrabackup备份工具。该工具官方下载地址为https://www.percona.com/software/mysql-database/percona-xtrabackup，它包含两个备份工具 —— xtrabackup和innobackupex。如果数据引擎为myisam或者个别表的数据引擎为myisam，则需要使用innobackupex。

在本例中，我们将使用mysqldump作为备份工具。

知识点二：date命令

date在Linux系统里可以显示当前的系统日期和时间。示例命令如下：

[root@aminglinux ~]# date

2018年 09月 15日 星期六 20:36:37 CST

在shell脚本中，date命令非常有用，就比如本例，要求备份文件名称要带有日期，我们就可以用date命令来实现，示例命令如下：

[root@aminglinux ~]# date +%F

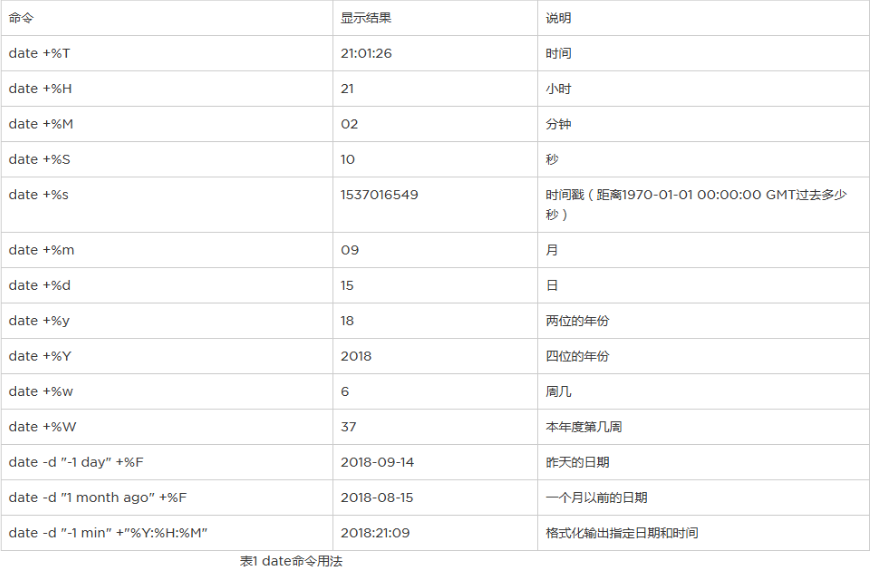
2018-09-15

对于本例的第7点要求，也可以用date的日期来实现，示例命令如下：

[root@aminglinux ~]# date +%d

15

下面我列出一些date命令常见的用法，如表1所示。



知识点三：找出一天以前的文件

在Linux系统里找文件有一个很厉害的工具 — find，从字面意思上看就知道这个工具就是用来“查找”符合要求文件的。下面这几种用法你会经常在shell脚本中使用：

1）找到logs目录下一周以前的老文件

>find logs/ -type f -mtime +7

2）找到/etc/目录下所有文件名以.cnf结尾的文件

find /etc/ -type f -name "\*.cnf"

3）找到当前目录下所有权限为777的目录

find . -type d -perm 777

4）找到/tmp/下大小超过1MB的文件

find /tmp/ -size +1M

5）找到当前目录下10天以前的文件并删除

find . -type f -mtime +10 -exec rm -f {} \;

还可以使用管道+xargs命令的形式：

find . -type f -mtime +10 |xargs rm

本例中，要求我们把一天以前的备份文件找到，可以这样做：

find /data/backup/ -name "\*.sql" -mtime +1

知识点四：gzip压缩

gzip支持压缩文件，但不支持压缩目录，压缩一个文件的命令是：

gzip file

此时，该文件被压缩为file.gz，也就是说该文件被压缩后也会被删除掉。解压.gz的文件，使用gzip命令的-d选项，如：

gzip -d file.gz

在Linux系统里，常见的压缩工具除了gzip外，还有bzip2,xz,zip等，其中bzip2和xz用法和gzip类似，而zip支持压缩目录。

结合知识点三和知识点四，把一天以前的备份文件压缩，命令为：

find /data/backup/ -name "\*.sql" -mtime +1 |xargs gzip

知识点五：rsync

rsync是一个远程同步数据的工具，多用在备份脚本中。rsync同步数据有两种方式，一种是通过ssh协议，需要知道对方机器用户密码或者通过密钥认证，此方式相对来说安全性稍差，毕竟知道了对方机器的密码或者建立了密钥认证，那么就能登录到对方机器上去，并可以执行命令。另外一种方式是通过在远程机器上开启一个rsyncd的服务，通过它特有的通信协议进行数据传输，该方式不用密码或建立密钥认证，但可以设置虚拟用户以及密码，这样就安全多了。本例中给出的远程备份地址，属于第二种方式。

rsync同步一个目录，最常用的命令为：

rsync -av /source\_dir/ user@ip::module\_name/dest\_dir/

rsync有几个常用的选项，如下：

-a 归档模式，表示以递归方式传输文件，并保持所有属性，等同于-rlptgoD, -a选项后面可以跟一个 --no-OPTION 这个表示关闭-rlptgoD中的某一个例如 -a--no-l 等同于-rptgoD

-r 对子目录以递归模式处理，主要是针对目录来说的，如果单独传一个文件不需要加-r，但是传输的是目录必须加-r选项

-v 打印一些信息出来，比如速率，文件数量等

-l 保留软链结

-L 向对待常规文件一样处理软链结，如果是SRC中有软连接文件，则加上该选项后将会把软连接指向的目标文件拷贝到DST

-p 保持文件权限

-o 保持文件属主信息

-g 保持文件属组信息

-D 保持设备文件信息

-t 保持文件时间信息

--delete 删除那些DST中SRC没有的文件

--exclude=PATTERN 指定排除不需要传输的文件，等号后面跟文件名，可以是万用字符模式（如\*.txt）

--progress 在同步的过程中可以看到同步的过程状态，比如统计要同步的文件数量、同步的文件传输速度等等

-u 加上这个选项后将会把DST中比SRC还新的文件排除掉，不会覆盖

选项确实有点多，不过不用担心，我工作这么多年，常用的选项页仅仅那么几个: (-a -v --delete --exclude )。

下面针对这些选项做一些列小实验：

1） 准备工作（建立目录以及文件）:

# mkdir rsync

# cd rsync

# mkdir test1

# cd test1

# touch 1 2 3

# ln -s /root/123.txt ./123.txt

# ls -l

总用量 0

-rw-r--r-- 1 root root 0 6月 10 12:58 1

lrwxrwxrwx 1 root root 13 6月 10 12:59 123.txt -> /root/123.txt

-rw-r--r-- 1 root root 0 6月 10 12:58 2

-rw-r--r-- 1 root root 0 6月 10 12:58 3

# cd ..

2）使用 -a 选项

# rsync -a test1 test2

# ls test2

test1

# ls test2/test1/

1 123.txt 2 3

这里有一个问题，就是本来想把test1目录直接拷贝成test2目录，可结果rsync却新建了test2目录然后把test1放到test2当中。为了避免这样的情况发生，可以这样做:

# rm -rf test2

# rsync -a test1/ test2/

# ls -l test2/

总用量 0

-rw-r--r-- 1 root root 0 6月 10 12:58 1

lrwxrwxrwx 1 root root 13 6月 10 12:59 123.txt -> /root/123.txt

-rw-r--r-- 1 root root 0 6月 10 12:58 2

-rw-r--r-- 1 root root 0 6月 10 12:58 3

加一个斜杠就好了，所以建议你在使用rsync备份目录时要养成加斜杠的习惯。在上面讲了-a选项等同于-rlptgoD，而且 -a 还可以和 --no-OPTIN 一并使用。下面看看-l选项的作用:

# rsync -av --no-l test1/ test2/

sending incremental file list

created directory test2

./

1

skipping non-regular file "123.txt"

2

3

sent 200 bytes received 72 bytes 544.00 bytes/sec

total size is 13 speedup is 0.05

使用-v选项看来就是方便，上例告诉我们跳过了非普通文件123.txt，其实123.txt是一个软连接文件，如果不使用-l选项则不理会软连接文件的。虽然加上-l选项会把软连接文件给拷贝过去，但是软连接的目标文件却没有拷贝过去，有时候咱们指向拷贝软连接文件所指向的目标文件，那这时候该怎么办呢？

3）使用-L选项

# rsync -avL test1/ test2/

sending incremental file list

created directory test2

./

1

123.txt

2

3

sent 231 bytes received 91 bytes 644.00 bytes/sec

total size is 0 speedup is 0.00

# ls -l test2/

总用量 0

-rw-r--r-- 1 root root 0 6月 10 12:58 1

-rw-r--r-- 1 root root 0 6月 10 12:39 123.txt

-rw-r--r-- 1 root root 0 6月 10 12:58 2

-rw-r--r-- 1 root root 0 6月 10 12:58 3

加上 -L 选项就可以把SRC中软连接的目标文件给拷贝到DST.

4） 使用-u选项

首先查看一下test1/1 和test2/1的创建时间（肯定是一样的），然后使用touch修改一下test2/1的创建时间（此时test2/1要比test1/1的创建时间晚了一些），如果不加-u选项的话，会把test2/1的创建时间变成和test1/1的创建时间一样。这样讲也许你会迷糊，不妨看一看:

# ll test1/1 test2/1

-rw-r--r-- 1 root root 0 6月 10 12:58 test1/1

-rw-r--r-- 1 root root 0 6月 10 12:58 test2/1

两者之间的创建时间是一样的，下面修改test2/1 的创建时间，然后不加-u同步:

# touch test2/1

# ll test2/1

-rw-r--r-- 1 root root 0 6月 10 13:20 test2/1

# rsync -a test1/1 test2/

# ll test2/1

-rw-r--r-- 1 root root 0 6月 10 12:58 test2/1

test2/1 的创建时间又变成和test1/1的创建时间一样了。下面加上 -u 再看看结果是怎么样的:

# touch test2/1

# ll test2/1

-rw-r--r-- 1 root root 0 6月 10 13:31 test2/1

# rsync -avu test1/ test2/

sending incremental file list

./

123.txt -> /root/123.txt

sent 100 bytes received 18 bytes 236.00 bytes/sec

total size is 13 speedup is 0.11

[root@localhost rsync]# ll test2/1

-rw-r--r-- 1 root root 0 6月 10 13:31 test2/1

ll test1/1

-rw-r--r-- 1 root root 0 6月 10 12:58 test1/1

加上-u 选项后，不会再把 test1/1 同步为 test2/1了。

5）使用 --delete 选项

首先删除test1/123.txt:

# rm -f test1/123.txt

# ls test1/

1 2 3

然后把test1/ 目录 同步到 test2/ 目录下:

# rsync -av test1/ test2/

sending incremental file list

./

1

sent 94 bytes received 34 bytes 256.00 bytes/sec

total size is 0 speedup is 0.00

[root@localhost rsync]# ls test2/

1 123.txt 2 3

test2/目录并没有删除掉123.txt, 下面加上 --delete 选项:

# rsync -av --delete test1/ test2/

sending incremental file list

deleting 123.txt

sent 52 bytes received 12 bytes 128.00 bytes/sec

total size is 0 speedup is 0.00

[root@localhost rsync]# ls test2/

1 2 3

test2/ 目录里的123.txt也被删除了，这就是 --delete 选项的用处。还有一种情况就是如果在DST增加文件了，而SRC当中没有这些文件，同步时加上 --delete 选项后同样会删除新增的文件:

# touch test2/4

# ls test1/

1 2 3

# ls test2/

1 2 3 4

# rsync -a --delete test1/ test2/

# ls test1/

1 2 3

# ls test2/

1 2 3

6）使用 --exclude 选项

# touch test1/4

# rsync -a --exclude="4" test1/ test2/

# ls test1/

1 2 3 4

# ls test2/

1 2 3

另外还可以使用匹配字符 \*

# touch test1/1.txt test1/2.txt

# ls test1/

1 1.txt 2 2.txt 3 4

# rsync -a --progress --exclude="\*.txt" test1/ test2/

sending incremental file list

./

4

0 100% 0.00kB/s 0:00:00 (xfer#1, to-check=0/5)

sent 104 bytes received 34 bytes 276.00 bytes/sec

total size is 0 speedup is 0.00

# ls test2/

1 2 3 4

上例中，也连带着使用了--progress 选项，这个主要是用来观察rsync同步过程的状态的。最后简单总结一下，平时你使用rsync同步数据的时候，使用-a选项基本上就可以达到我们想要的效果了，只是有时候会有个别的需求，会用到 -a --no-OPTION, -u, -L, --delete, --exclude 以及--progress 这些选项，还有些选项没有介绍，如果在以后的工作中遇到特殊需求了，就去查一下rsync的man文档吧。

本案例参考脚本

有了以上的背景知识点，再去写最终的shell脚本就很容易了。下面是我写的一个参考脚本，内容如下：

#!/bin/bash

##该脚本用来备份本机数据库

##本地保留一周，远程保留一个月

##作者：

##日期：2018-09-15

##版本：v0.1

mysqldump="/usr/local/mysql/bin/mysqldump"

bakuser="backup"

passwd="34KpmyzUq"

bakdir="/data/backup"

remote\_dir="rsync://10.10.20.100/mysqlbak"

d1=`date +%F`

d2=`date +%d`

#定义日志

exec &>; /tmp/mysql\_bak.log

echo "mysql backup begin at `date`"

#对所有数据库进行遍历

for db in db1 db2 db3 db4 db5

do

$mysqldump -u$bakuser -p$passwd $db >$bakdir/$db-$d1.sql

done

#对1天前的所有sql文件压缩

find $bakdir/ -type f -name "\*.sql" -mtime +1 |xargs gzip

#查找一周以前的老文件，并删除

find $bakdir/ -type f -mtime +7 |xargs rm

#把当天的备份文件同步到远程

for db in db1 db2 db3 db4 db5

do

rsync -a $bakdir/$db-$d1.sql $remote\_dir/$db-$d2.sql

done

echo "mysql backup end at `date`"

脚本中，有几个点需要解释一下。

1）脚本第一行#!/bin/bash是固定格式，用来指定解释器bash的具体位置，前面的"#!"是必须要加上的。

2）所有以”#”开头的行，都是解释说明的文字，为了便于大家理解，所以我都用了中文，建议大家尽可能地用英文，因为中文可能会出现乱码。这些注释文字用来对该脚本的描述，通常会写脚本的作用、作者以及时间或者版本等信息。

3）脚本中我用了大量的变量，这是因为这些信息可能会改变，把它们定义到变量里方便修改。

4）shell脚本中，可以使用反引号来引用一条命令的结果，让后将其保存到变量中

5）&>;指定正确输出和错误输出到/tmp/mysql\_bak.log

6）在shell脚本中一旦遇到了exec命令，则它会接管该shell脚本中后续所有的命令，我们可以理解为从exec那行以下所有的命令都“被拉到了一个盒子里去执行”，而这个“盒子”就是exec本身，所以exec &>; 1.txt，就可以理解为这个“盒子”里所有的操作的输出全部都到1.txt里。

7）shell脚本中循环遍历，可以使用for，语法是for ... ; do ...; done

8）本例中要求远程备份文件保存1个月，由于脚本不能到远程机器上执行find命令，所以在这我想了一个“曲线救国”的方法，date +$d表示日期，一个月的所有日期都有了，每天一个新sql文件。例如，本月15日sql文件将会覆盖上个月15日的，同理，下个月会覆盖本月的相同日期的文件，因为文件名是一样的。但无论如何，总能保存一个月。

9）本例中，我并没有考虑要把当天的sql文件压缩后再同步到远程机器上，如果想要压缩，也可以这样做：a）压缩；b）同步；c）解压缩。