## 案例六：检测两台机器上文件的差异

生产环境中，web服务器大多会做负载均衡，所以有多台机器上跑着同样的web程序代码。如果严格按照规范流程上线，即不人为特意更改代码，那么这多台机器上的代码一定是一样的，并不会存在两台机器上同一个文件内容不同的情况。但，本案例的需求就是要检查两台机器同一个文件的差异，毕竟我们不确定服务器上的代码是否有人为改动过。具体需求如下：

1）两台机器A和B，检查目标目录为/data/wwwroot/www.abc.com/，路径一致

2）需要过滤目录uploads、tmp两个目录，即这两个目录下的文件不需要对比差异

3）以A机器上的文件作为标准，B机器少了文件和改了文件需要记录，多了文件不用考虑

4）假设A机器可以免密码登录B机器

5）把有差异和丢失的文件列表存入文件/data/change.log

知识点一：A机器免密码登录B机器

这里所谓的免密码登录指的是，A机器可以直接ssh B机器，不用输入密码。实现免密码登录的方法有两种，一个是使用expect脚本（这部分内容后面的案例中会涉及到，所以此处不再详细介绍），另外一种方法是通过密钥认证，具体操作步骤是这样的。

首先需要在A机器上生成一对密钥，其中包含私钥和公钥，私钥放在A机器上，公钥放到B机器上，私钥用来加密，公钥用来解密。生成密钥对的命令如下：

# ssh-keygen //执行后一直按回车即可

Generating public/private rsa key pair.

Enter file in which to save the key (/root/.ssh/id\_rsa): //可以在这里更改密钥保存的路径，如果按回车则存放到默认路径下。

Created directory '/root/.ssh'.

Enter passphrase (empty for no passphrase)://这个地方需要输入密钥的密码，直接按回车则密钥密码为空。

Enter same passphrase again: //这里是密码确认

Your identification has been saved in /root/.ssh/id\_rsa.

Your public key has been saved in /root/.ssh/id\_rsa.pub.

说明：因为当前用户是root，所以会在root家目录/root/下生成一个.ssh目录，密钥对在.ssh目录下。

这样就会在~/.ssh/目录下生成了两个文件id\_rsa和id\_rsa.pub，其中id\_rsa为私钥，id\_rsa.pub为公钥。如果机器上已经存在了这样的一对文件，那可以直接使用它们了。 当然，也可以指定文件的名字，如下：

# ssh-keygen -f /tmp/123 //这样会在/tmp/下生成123和123.pub两个文件

生成密钥对后，需要把公钥传到目标机器上，假设B机器的IP为192.168.0.110，则传输公钥的方法是：

# ssh-copy-id root@192.168.0.110 //这里的root@可以省略掉，即只写IP。这个操作，需要我们手动输入对方机器目标用户（root）的密码

上面这个命令会把~/.ssh/id\_rsa.pub文件内容写到目标机器目标用户（如，root）家目录下的.ssh/authorized\_keys文件里，注意这里是追加写入，也就是说如果.ssh/authorized\_keys文件里有内容，它不会覆盖，而是追加。若想指定特定的私钥呢？比如，上例中那个/tmp/123，命令是这样的：

# ssh-copy-id -i /tmp/123 root@192.168.0.110 //这里的-i选项后面指定私钥文件路径，它可以自动产生公钥

做完上面的两步，就可以用ssh命令登录目标机器了，而且也不用输入密码。它默认会找~/.ssh/id\_rsa这个私钥，但如果使用的是另外的私钥文件，还需要在ssh时指定文件路径，如下：

# ssh -i /tmp/123 192.168.0.110

知识点二：find中过滤指定目录

先来看一个简单的需求，要求查找当前目录下除了123目录（./123）外，所有的文件，有两种方法。

方法一：

# find ./ -type f |grep -v '^\./123'

说明：这里首先find出所有的文件列表，然后再针对结果用grep排除掉123目录。对于此方法，简单的场景是没问题的，如果某个文件的文件名中包含了123关键词，就不太好处理了。

方法二：

# find ./ -path './123\*' -prune -o -type f

说明：-path类似于shell中的正则匹配，-prune -o 组合在一起使用，会把-prune前面的匹配排除掉，如果是要排除多个目录，可以这样用：

# find ./ \( -path './123\*' -o -path './abc\*' \) -prune -o -type f

注意：-path后面跟的那个目录名字后面不要加/，否则就不正确了。

知识点三：对比两个文件是否有差异

可以使用diff命令，当然也可以对比两个文件的md5sum值。diff命令能比较出差异来，并且会把具体差异的内容显示出来，在本例中并不需要知道具体差异内容，而且我们没有办法用diff对比本机和远程机器上的两个文件，所以只能使用md5sum。先来看一下diff命令的用法：

# echo -e "123\n123" >2.txt

# echo -e "123\nabc" >1.txt

# cat 2.txt

123

123

# cat 1.txt

123

abc

# diff 1.txt 2.txt

2c2

< abc

---

> 123

说明：echo -e可以识别换行\n，diff出来的结果，2c2指的是第二行有差异，<表示左边的文件，>表示右边的文件

而md5sum命令这样使用：

# md5sum 2.txt

d41d8cd98f00b204e9800998ecf8427e 2.txt

前面这个字符串，就是2.txt文件的md5值，如果我们修改一下2.txt，则md5值会发生改变：

# echo "123" >> 2.txt

# md5sum 2.txt

ba1f2511fc30423bdbb183fe33f3dd0f 2.txt

要想判断两个文件是不是一样，只要对比两个文件的md5值即可。

知识点四：嵌入文档（Here Document）

嵌入文档，英文名叫做Here Documents，大家可能见过，但恐怕并不知道这个专业术语，先来上一段shell代码：

#!/bin/bash

cat > 1.txt <<EOF

Hello

My

name

is aming.

This's a test text.

EOF

运行这个脚本后，会产生1.txt文档，内容为：

Hello

My

name

is aming.

This's a test text.

这个用法就是Here Documents。这里的&quot;EOF&quot;叫做标识符，还可以换成其他字符串，比如写成ABC、123等等，随便你定义成啥，但要保证前后对应，也不要造成和其他字符串混淆。通常大家习惯写EOF，这样也容易让别人识别。就跟“shell脚本名字要以.sh结尾”同样的道理。</p>

Here Documents需要注意一点，最后面那个标识符（EOF）必须要顶格写，当然，不一定非得 cat << EOF，你也可以换成其他命令，例如

wc -l <<EOF

1

2

3

EOF

知识点五：while循环遍历文件

在shell中我使用for循环比较多，但用for循环遍历一个文件内容时，也就是说按行来遍历时就会有一些问题。因为for循环遍历对象以空白字符或者换行作为分割符，例如如下文本用for循环则无法实现预期：

# vim test.txt //内容如下

ab 1

123

234

# for s in `cat test.txt`; do echo $s; done //命令行下写shell脚本，用;分割多行

ab

1

123

234

而使用while循环，效果是这样的：

# cat test.txt|while read s ; do echo $s; done //这里的s是一个变量，可以自定义

ab 1

123

234

所以，要想遍历一个文件所有行时，可以用while read循环来实现。

本案例参考脚本

结合以上的五个知识点，下面我给出最终的参考脚本，内容如下：

#!/bin/bash

##对比两台机器上的文件差异

##作者：

##日期：2018-09-27

##版本：v0.2

#假设B机器IP为192.168.0.110

B\_ip=192.168.0.110

dir=/data/wwwroot/www.abc.com

#首先检查/tmp/md5.list文件是否存在，存在的话就删除掉，避免影响后续操作

[ -f /tmp/md5.list ] && rm -f /tmp/md5.list

#把除了uploads以及tmp目录外其他目录下的全部文件列出来

cd $dir

find . \( -path "./uploads\*" -o -path "./tmp\*" \) -prune -o -type f > /tmp/file.list

#用while循环，求出所有文件的md5值，并写入一个文件里

cat /tmp/file.list|while read line

do

md5sum $line

done > /tmp/md5.list

#将md5.list拷贝到B机器

scp /tmp/md5.list $B\_ip:/tmp/

#判断/tmp/check\_md5.sh文件是否存在

[ -f /tmp/check\_md5.sh ] && rm -f /tmp/check\_md5.sh

#用Here Document编写check\_md5.sh脚本内容

cat >/tmp/check\_md5.sh << EOF

#!/bin/bash

dir=/data/wwwroot/www.abc.com

##注意，这里涉及到的特殊符号都需要脱义，比如反引号和$

n=\`wc -l /tmp/md5.list|awk '{print \$1}'\`

for i in \`seq 1 \$n\`

do

file\_name=\`sed -n "\$i"p /tmp/md5.list |awk '{print \$1}'\`

md5=\`sed -n "\$i"p /tmp/md5.list|awk '{print \$2}'\`

if [ -f \$file\_name ]

then

md5\_b=\`md5sum \$file\_name\`

if [\$md5\_b != \$md5 ]

then

echo "\$file\_name changed."

fi

else

echo "\$file\_name lose."

fi

done > /data/change.log

EOF

scp /tmp/check\_md5.sh $B\_ip:/tmp/

ssh $B\_ip "/bin/bash /tmp/check\_md5.sh"

说明：本脚本的思路是这样的：

1）在A机器上遍历所有文件，对每一个文件计算md5值存入一个临时文件里

2）把临时文件拷贝到B机器，留着备用

3）编写在B机器上要执行的对比md5值的脚本

4）将脚本传到B机器

5）执行脚本

而在B机器上执行的脚本，思路是：

1）根据md5值临时文件里的文件列表，逐一计算B机器上文件的md5值

2）对比B机器上文件的md5值是否和A上的一样，同时判断该文件是否存在