1、 用 SSH 实现无密码自动登录

SSH 广泛用于脚本自动化。借助 SSH，我们可以在远程主机上执行命令并读取数据。SSH 使用用户名和密码进行认证，在 SSH 命令的执行过程中提示输入密码，但是在自动化脚本中，SSH 命令可能在一个循环中执行上百次，每次都得提供密码的话，显然不实际。因此，我们需要将登录过程自动化。SSH 就包含一个内置的特性，可以使用 SSH 密钥实现自动登录。[1]

SSH 采用公钥和私钥的非对称加密技术进行自动化认证。我们可以通过 **ssh-keygen** 命令创建认证密钥。要想实现自动化认证，**公钥**必须放在**服务器**中（将其**加入文件 ~/.ssh/authorized_key**），与公钥对应的**私钥**应该放在你用来登录的**客户机~/.ssh** 目录中。另一些与 SSH 相关的配置信息（例如，authorized_keys 文件的路径与名称）可以通过修改文件/etc/ssh/sshd_config 进行配置。[1]

（1） 在**客户机**上生成密钥，输入命令后一直回车即可。

```
[root@host ~]# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
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:
SHA256:C5yJUI3rRyCqaSA32x+jyYDOV9iijMUHt6qGIMVJbLY root@host.localdomain
The key's randomart image is:
+---[RSA 2048]----+
|  .   .o         |
|   *.o .         |
|  *.+ o          |
| + E.oo.o        |
|+* B+o= S        |
|*.=o=o+. .       |
|X..=o= o.        |
|o*..+ .          |
|o..              |
+----[SHA256]-----+
```

执行效果 1-1 创建密钥对

仅仅输入一行命令来生成一对公钥和私钥（就是一路回车这种行为）是不安全的，但如果你输入了口令生成的话，**在编写脚本时也需要输入相应的口令**，出于便利的目的这里就不输入口令了。（已测试，确实如此）

(2) 前一步在~/.ssh 目录下生成了私钥文件（id_rsa）和公钥文件（id_rsa.pub），然后我们需要把客户机上的公钥添加到远程服务器的 **~/.ssh/authorized_keys** 文件中。使用以下命令添加公钥。

```
ssh USER@HOST -p PORT "cat >> ~/.ssh/suthorized_keys" < ~/.ssh/id_rsa.pub
Password:（注意，这个命令需要输入密码）
```

这样就设置好了密钥认证，从现在开始 SSH 在运行过程中就不会提示输入密码了。

```
ssh USER@HOST -p PORT uname
Linux
```

2、 用 SSH 在远程主机上运行命令

SSH 是一个很有意思的系统管理工具，它能够通过 shell 登录并控制远程主机。SSH 是 Secure Shell 的缩写。我们可以登录到远程主机上来执行 shell 命令，就好像是在本低主机上执行这些命令一样。SSH 使用加密通道来传输数据。[1]下面将介绍在远程主机上运行命令的各种方法。

SSH 采用交互方式询问用户密码，一旦认证成功，将会返回一个 shell。SSH 服务器默认在 22 端口运行，不过有些 SSH 服务器并不在这个端口上运行。针对这种情况，需要使用**-p PORT** 的方式指定端口。[1]

```
ssh USER@HOST -p PORT
```

如果需要将任务自动化，那么很多时候我们并不需要开启一个长久的 shell 进行交互，我们只需要通过 ssh 在远程 shell 上执行若干命令。因为 SSH 是询问用户密码的交互方式，所以如果想要实现自动化脚本，需要对 SSH 进行自动登录配置。

2.1、 命令格式

通过 SSH 远程执行命令的语法如下：

```
ssh USER@HOST -p PORT 'COMMANDS'
```

也可以输入多条命令，在命令之间以分号进行分隔：

```
ssh USER@HOST -p PORT 'command1; command2; command3'
```

2.2、 使用标准输入和标准输出

可以用 stdin 标准输入来给 ssh 传入远程命令，也可以通过 stdout 标准输出来获取命令的输出。

```
    # 标准输入输入命令
    ssh USER@HOST -p PORT < command.txt
    # 标准输出获取命令输出
    ssh USER@HOST -p PORT 'COMMANDS'> stdout.txt 2>errors.txt
    # 标准输入将管道内容作为命令序列输入
    cat command.txt | ssh USER@HOST -p PORT> stdout.txt 2>
errors.txt
```

2.3、 编写远程操作脚本

同时，也可以使用 shell 脚本，对远程服务器进行批量操作，下面编写一个用来收集远程主机运行时间（uptime）的脚本：

```
#!/bin/bash
#文件名：uptime.sh
#用途：系统运行时间监视器

IP_LIST="192.168.0.1 192.168.0.2 192.168.0.3"
USER="root"
PORT="28"

for IP in $IP_LIST;
do
    utime=$(ssh $USER@$IP -p $PORT uptime | awk '{ print $3 }')
    echo $IP uptime:  $utime
done
```

输出应该是：

```
$ ./uptime.sh
192.168.0.1 uptime: 8:05,
192.168.0.2 uptime: 8:05,
192.168.0.3 uptime: 8:05,
```

2.4、 压缩命令

SSH 协议支持对数据进行压缩传输，当带宽有限时，这一功能很方便。使用 ssh 命令的选项-C 启用压缩功能：[1]

```
    ssh -C USER@HOST -p PORT 'COMMANDS'
```

2.5、 将数据重定向至远程 shell 命令的 stdin

用 SSH 实现无密码自动登录中，有一步操作是将客户机上的公钥上传至远程服务器，使用了如下命令：

```
ssh USER@HOST -p PORT "cat >> ~/.ssh/suthorized_keys" < ~/.ssh/id_rsa.pub
```

这就是将数据通过本地的 stdin 重定向到远程 shell 命令的 stdin 的一种用法，相同里的例子还有：

```
    echo file | ssh USER@HOST -p PORT 'cat >> list'
```

或

```
    ssh USER@HOST -p PORT 'cat >> list' < file
```

这两种方式，前者是通过管道，后者直接使用 stdin，将文件内容重定向到远程命令 cat 的 stdin 中去，效果就是本地 file 的内容被写入到远程的 list 文件中去。

3、 参考资料

[1] 《Linxu Shell 脚本攻略》-Shantanu Tushar [2013-12-25]