

# SSH和服务入门

前言：

终于到了这一节了，万众瞩目的SSH终于登场了。这下我们可以远程登录Linux了，这个还是挺重要的。因为在企业里，我们就是要远程登录远处的机房里的服务器，然后再在此基础上操作，这时候就要SSH了。

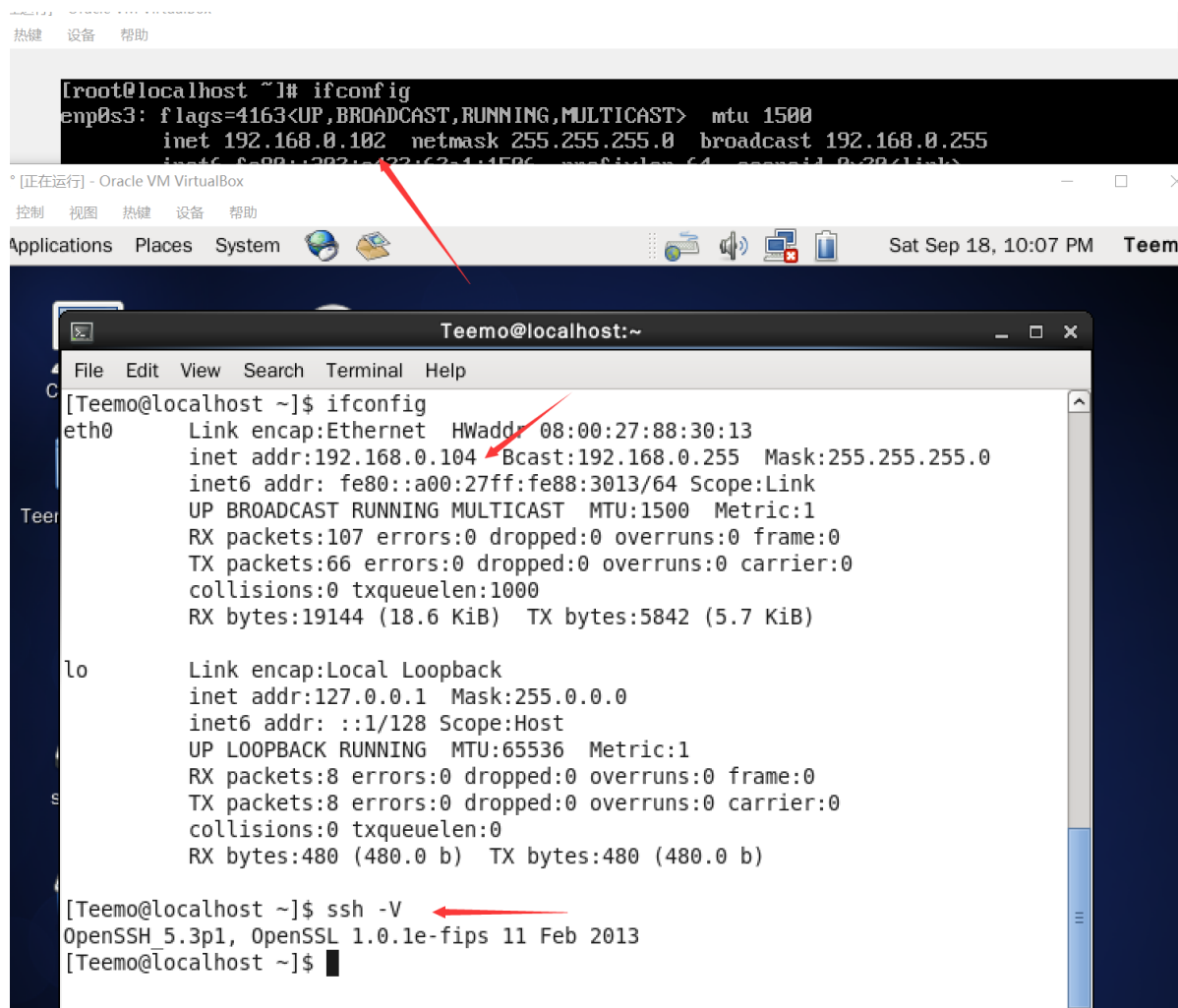
正文：

## 1、C/S模型

C是client，S是server，即是指客户端和服务端模型。软件能分出C和S两端，然后装在两边的计算机上，就形成了C/S模型。而我们即将要讲的SSH就是C/S模型。不光如此，Linux上的绝大多数提供服务的软件都是C/S模型的。

## 2、准备SSH客户端

如果你是mac用户，那么直接在应用里搜索一个叫terminal的软件即可，它很像Linux的命令行，这是因为Linux和mac都是类UNIX的系统。如果你是Windows，乖乖再装一台虚拟机当作SSH客户端吧。



如图，我本来就有两台机器。我发现它们的IP不一样。然后 ssh -V 确认SSH客户端。

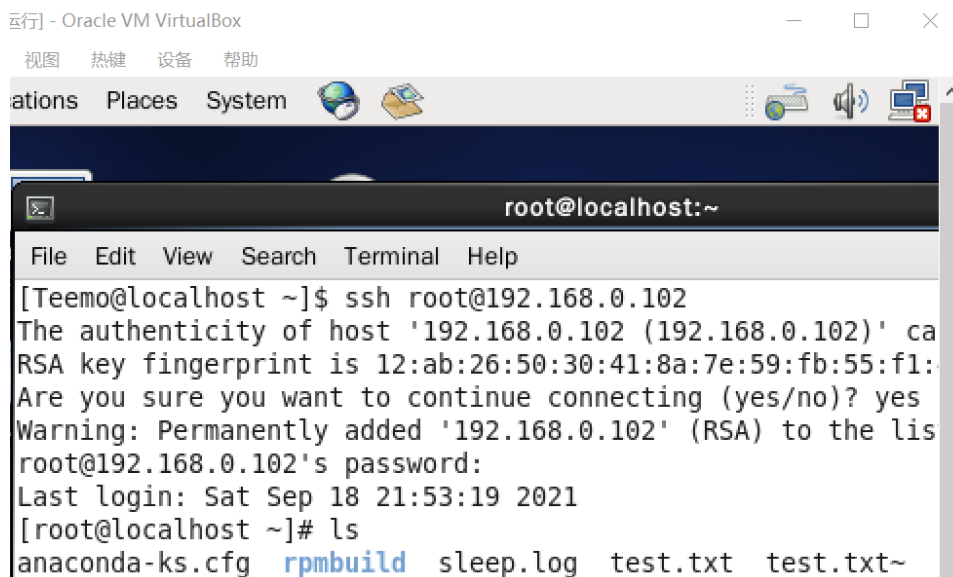
## 3、异地登录不是梦

ssh + 用户名@IP地址，即可实现远程登录账户：

```
[root@localhost ~]# ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.102 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::302:c422:62a1:1506 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:ec:39:33 txqueuelen 1000 (Ethernet)
    RX packets 199 bytes 26571 (25.9 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 112 bytes 9855 (9.6 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 68 bytes 5912 (5.7 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 68 bytes 5912 (5.7 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

```
[root@localhost ~]# ls
anaconda-ks.cfg  rpmbuild  sleep.log  test.txt  test.txt~
```



我使用了ssh客户端用虚拟机，成功登录了ssh服务端，你也来试试吧——最开始它会让你验证，输入密码即可。现在在办公室就可以轻松的操控远方大型机房里的那些沉甸甸的服务器了，好事。

#### 4、结合SSH谈用户管理

```

root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
systemd-network:x:192:192:systemd Network Management:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
polkitd:x:999:998:User for polkitd:/:/sbin/nologin
libstoragemgmt:x:998:997:daemon account for libstoragemgmt:/var/run/lsm:/sbin/nologin
abrt:x:173:173:/:etc/abrt:/sbin/nologin
rpc:x:32:32:Rpcbind Daemon:/var/lib/rpcbind:/sbin/nologin
apache:x:48:48:Apache:/usr/share/httpd:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
postfix:x:89:89:/:var/spool/postfix:/sbin/nologin
chrony:x:997:995:/:var/lib/chrony:/sbin/nologin
ntp:x:38:38:/:etc/ntp:/sbin/nologin
tcpdump:x:72:72:/:/sbin/nologin
usertest:x:1000:1000:/:home/usertest:/bin/bash
linuxuser:x:1001:1001:/:opt:/bin/bash

tss:x:59:59:Account used by the trousers package to sandbox the tcsd daemon:/dev/null:/sbin/nologin

```

可以修改用户所用的目录，这样我们远程登录就登录到了特定的文件夹里。可以尝试一下，然后pwd看路径。

如果不想被别人登录到某个用户，可以将用户后面的/bin/bash改成/sbin/nologin，字面意思，然后别人就不能登录这个用户了。现在可以尝试exit退出ssh服务，然后改成如此再试试登录，你会发现拒绝访问。

## 5、学习软件的一些概念

说到配置文件，那我们铁定不陌生了，刚刚用的passwd也是配置文件呢。

我们现在用的这些命令啊，服务啊之类的，都是rpm包来的，它们都算Linux下的软件，所以我们先区分一下以下一些概念：

- ①所有通过编程编译得来的，都可以称为一个软件；
- ②凡是在前台看得见，并且可以一次性快速被使用执行的软件，叫做命令或工具，如ls一类；
- ③凡是在后台默默打工的，并且可以被动地、持续不断地给客户端提供某种功能的软件，称作服务，例如SSH、HTTP一类；
- ④还存在一些深层的软件，高深莫测的，用一般方法难以看见的软件，如驱动、协议、内核等。

我们如果想有效地管理好一些服务为己所用，就需要掌握它们的配置文件，通过修改它们的配置文件，改变它们的服务方式。说白了，服务的操控，就是通过更改它们的配置，让它们变化出不同的形态和功能。

一般来说，一款服务在安装好后，它们的配置文件所在的路径都比较有规律，如/usr/local、/var/lib等。如果想找一个软件的配置文件，可以通过rpm -qa | grep 名称 来筛出软件的rpm包，然后找到对应的rpm包后，通过 rpm -ql 包全称 来找出软件的配置文件所在路径。

```
[root@localhost ~]# rpm -qa | grep openssh
openssh-7.4p1-21.el7.x86_64
openssh-clients-7.4p1-21.el7.x86_64
openssh-server-7.4p1-21.el7.x86_64
[root@localhost ~]# rpm -ql openssh-server-7.4p1-21.el7.x86_64
package openssh-server-7.4p1-21.el7.x86_64 is not installed
[root@localhost ~]# rpm -ql openssh-7.4p1-21.el7.x86_64
package openssh-7.4p1-21.el7.x86_64 is not installed
[root@localhost ~]# rpm -ql openssh-server-7.4p1-21.el7.x86_64
/etc/pam.d/ssh
/etc/ssh/ssh_config ←
/etc/sysconfig/ssh
/usr/lib/systemd/system/ssh-keygen.service
/usr/lib/systemd/system/ssh.service
/usr/lib/systemd/system/ssh.socket
/usr/lib/systemd/system/ssh@.service
/usr/lib64/fipscheck/ssh.hmac
/usr/libexec/openssh/sftp-server
/usr/sbin/ssh
/usr/sbin/ssh-keygen
/usr/share/man/man5/moduli.5.gz
/usr/share/man/man5/ssh_config.5.gz
/usr/share/man/man8/sftp-server.8.gz
/usr/share/man/man8/ssh.8.gz
/var/empty/ssh
[root@localhost ~]# _
```

如图，config文件就是ssh的配置文件，别像我一样把i打成l，结果搞半天没东西。打开它看看吧。

一般来说，配置文件里首先会有一大段以#开头的英文，这些就是我们所说的注释了——单纯给你解释这个文件的用途。随后的关键内容的格式都比较统一了，要么 key/value（键值形式）、要么变量定义形式（变量名=值）、要么其他形式。除此之外，配置文件中可能还有很多配置行被屏蔽了，可以根据需要去掉前面的#号来启动它们。

## 6、着重讲讲SSH的配置文件中的关键5行

以后我们会接触各种各样的服务，它们的配置文件内容多的数不清，所以我的学习要在实践的基础上，记忆那些常用的配置项，至于别的，需要用到时再查询相关文档学习。接下来我们讲SSH必掌握的五条。

①ListenAddress 0.0.0.0：监听地址范围。一台服务器可以有多个网卡，这也就意味着它有相应的多个ip地址，我若是要用ssh登录这个账户，那么我该@后接哪个ip地址呢？0.0.0.0表示登录哪个ip都可以，当然你也可以自己设置一下，以限制ssh的登录，告诉外来登录者若要登录只能登录该ip，其余全拒绝访问。

vim /etc/ssh/ssh\_config 打开配置文件，修改ip后保存退出，然后CentOS6下 service sshd restart 重启服务、CentOS7下 systemctl restart sshd 重启服务，使得服务配置更新。

②Port 22：端口号。可以把操作系统和上面运行的各种软件服务理解想象成一桌桌饭菜，客户端客人来了，先根据ip找到房间号（服务器），然后再根据端口号找到桌子号（服务所在），才能享用饭菜（服务）。一个操作系统上能提供的端口号有限的，一个号被占用了，另一个服务就不能使用这个号了。

SSH服务的默认端口号是22。

```

[root@localhost ~]# netstat -tnlp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:25           0.0.0.0:*               LISTEN      1339/master
tcp        0      0 0.0.0.0:111            0.0.0.0:*               LISTEN      760/rpcbind
tcp        0      0 0.0.0.0:22             0.0.0.0:*               LISTEN      1103/sshd
tcp6       0      0 :::1:25                :::*                     LISTEN      1339/master
tcp6       0      0 :::111                  :::*                     LISTEN      760/rpcbind
tcp6       0      0 :::22                   :::*                     LISTEN      1103/sshd
[root@localhost ~]#

```

netstat -tnlp 命令可以查看系统中正处于监听状态下的所有服务和它们对应的端口号，还有它们分别都监听在什么IP地址。

③PubkeyAuthentication yes：公匙验证，开启。后面会讲到这个。

④PasswordAuthentication yes：密码验证，开启。很好理解，输入账号密码验证SSH登录。

⑤PermitRootLogin yes：是否允许登录root账户。

## 7、配置SSH免密码的登录

上面我们提到一个公钥验证登录，开启后为的就是这个免密码安全登录。整个流程是这样的：

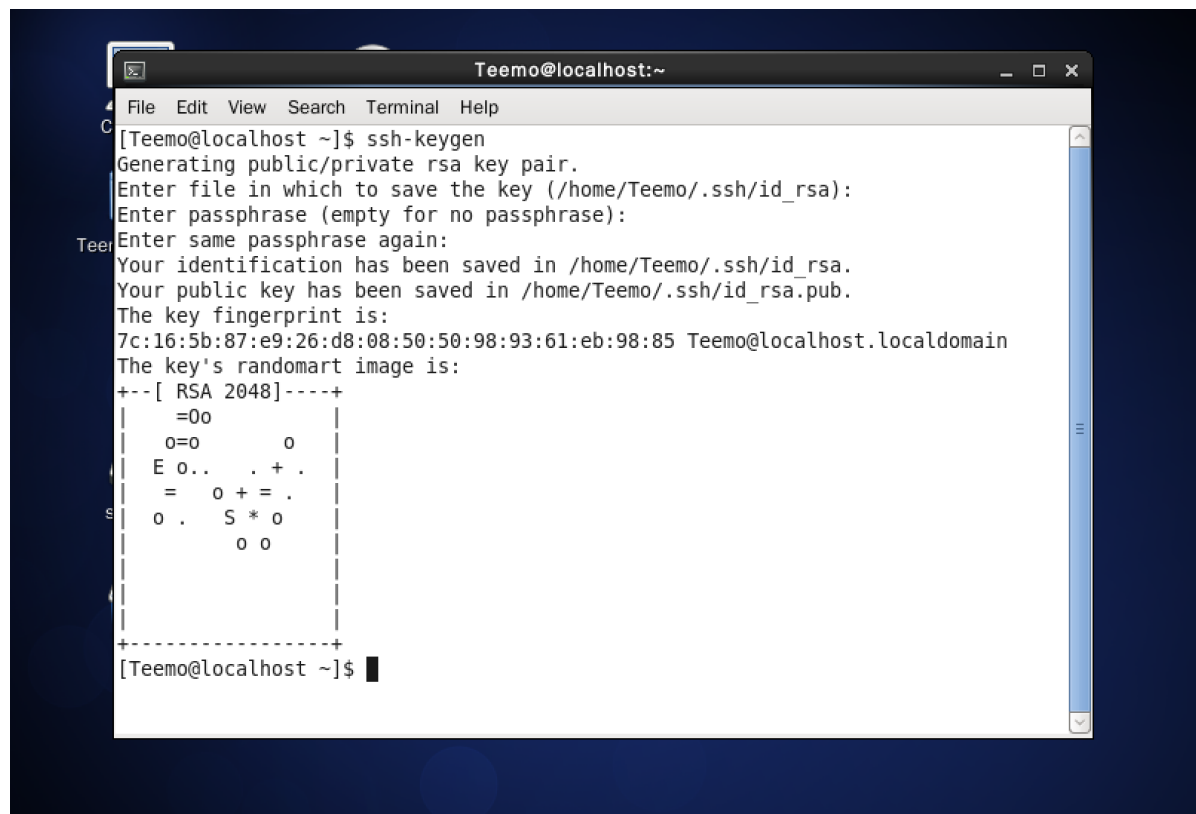
①SSH服务端要两把钥匙才能被打开（登录）。

②SSH客户端先生成了两把万能钥匙，一把私钥留给自己，一把公钥送给SSH服务端。

③SSH服务端每次验证登录时，会先变出一把锁，然后交给SSH客户端。注意，每次变出的锁都不一样的。

④SSH客户端把私钥插入锁里，然后交给SSH服务端，服务端插入自己的公钥，成功打开锁，登录验证成功。

接下来实操，配置公钥免密码登录：



```

Teemo@localhost:~
File Edit View Search Terminal Help
[Teemo@localhost ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/Teemo/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/Teemo/.ssh/id_rsa.
Your public key has been saved in /home/Teemo/.ssh/id_rsa.pub.
The key fingerprint is:
7c:16:5b:87:e9:26:d8:08:50:98:93:61:eb:98:85 Teemo@localhost.localdomain
The key's randomart image is:
+--[ RSA 2048 ]-----+
|      =0o              |
|      o=0             |
|    E 0.. . + .       |
|      =  o + = .       |
|    o .   S * o       |
|                      |
+-----+
[Teemo@localhost ~]$

```

输入 ssh-keygen 后一路Enter狂奔即可。这个命令执行后，它会在当前家目录下生成一个隐藏目录.ssh，然后 id\_rsa（私钥）和 id\_rsa.pub（公钥）放入其中。那上面有一句Enter passphrase，意思是给私钥再设置一个密码，但是我们都是免密码登录了，就没这个必要了。

```

[root@localhost ~]# mkdir .ssh
mkdir: cannot create directory '.ssh': File exists
[root@localhost ~]# ls -a
.          .bash_history  .bashrc  .cshrc  rpmbuild  .ssh      test.txt~
..         .bash_logout  .cache   .emacs.d .rpmmacros .tcshrc   .viminfo
anaconda-ks.cfg .bash_profile .config  .lessht  sleep.log  test.txt
[root@localhost ~]# ls -ld .ssh
drwx-----. 2 root root 25 Sep 18 22:21 .ssh
[root@localhost ~]# vim .ssh/authorized_keys_

```

接下来在服务端的root目录下创建一个隐藏文件夹.ssh，由于之前远程登录过，所以自动创建了。要是没有，则手动mkdir创建，并chmod 770 .ssh改变权限再chown -R root:root .ssh改变属主和属组。最后，在.ssh目录下创建一个authorized\_keys文件来存放公钥。

```

root@localhost:~
File Edit View Search Terminal Help
[Teemo@localhost ~]$ scp .ssh/id_rsa.pub root@192.168.0.102:~/.ssh/authorized_keys
root@192.168.0.102's password:
id_rsa.pub                                100% 409      0.4KB/s   00:00
[Teemo@localhost ~]$ ssh root@192.168.0.102
Last login: Sun Sep 19 11:11:27 2021 from 192.168.0.105
[root@localhost ~]# ls
anaconda-ks.cfg  rpmbuild  sleep.log  test.txt  test.txt~
[root@localhost ~]#

```

```

qy0KwNMLThp/3k85L1B11kyU7IzyvF0JYqRQC7ehscgBMmqWSIycuwRa2ne97UMdtzCTwxXDNbklr8rtZsXRDZgseYrIcbSjmcB
mUYuxWn139bB9g9St4mgBfU4bmkv1Y1T5iPr1S09YqhB0rGq0wIE4nNBXmaOrQEIEwXQdigrUt+brS4UnWjPbP9byE151oMF1bp
EC6Fb0NDhmWBMUrDLtb0qQQie2W9YMayjPC5BRKo4s8zPzIdQq5sa0GGZvNy/ojL+xyiHR84WJ+jQ== Teemo@localhost.loc
aldomain

```

如图，在客户端上用scp指令把id\_rsa.pub文件内容覆盖到服务端的authorized\_keys文件里，然后就可以免密码登录了。一大串密密麻麻的英文数字看麻了。

## 8、真正的SSH远程执行命令

上面我们说到的，其实是通过SSH服务登录到服务器的机子上，然后在服务器的机子上执行命令，所以相当于我们是“飞”到服务器那边了，那有没有一种真正的在原地的但还是能执行操作服务器的方式呢？

如果设备不同，hostname后会出现不同的机器名。然后在ssh 登录命令最后加一个“hostname”，注意是有双引号的哦，登录完成后hostname查看设备名称，你会发现没变。这种用法在编写脚本程序的时候会反复用到。

后记：

这波啊，这波是异地登陆。

