

GNU/Linux SSH



GNU/Linux SSH

SSH 服务器：

远程管理以前基本上都采用 Telnet. 但是其明文传输密码等问题都非常的令人头痛. 因此 SSH 的出现解决了许多在远程管理方式中产生的问题



GNU/Linux SSH

Secure Shell

SSH 采用了多种认证加密方式，解决了传输中数据加密和身份认证等问题，比较有效的防止网络嗅探与 IP 欺骗等问题



GNU/Linux SSH

Secure Shell

在 Linux/Unix 下目前仍然广泛采用的 SSH 是 OpenSSH 程序来实现 SSH 协议的。

OpenSSH 是由最著名的 Unix 中 OpenBSD 开发于维护的。



GNU/Linux SSH

ssh 配置文件

1. SSH 服务的配置文件位于
`/etc/ssh/sshd_config`
2. SSH 客户端配置文件位于
`/etc/ssh/ssh_config`



GNU/Linux SSH

sshd 配置文件概述：

1) 设置 SSH 服务监听的端口号 
Port 22

2) 设置使用 SSH 协议的顺序 
Protocol 2,1

3) 设置 SSH 服务器绑定的 IP 地址 
ListenAddress 0.0.0.0



GNU/Linux SSH

sshd 配置文件概述：

4) 是否允许 root 管理员登录 [最好不]

PermitRootLogin yes



5) 设置是否允许空密码用户登录

PermitEmptyPasswords on



6) 设置是否使用口令认证方式

PasswordAuthentication yes



GNU/Linux SSH

sshd 配置文件概述：

7) 是否允许用户使用 X 程序 
X11Forwarding yes

8) 是否允许用户使用 ssh 的子系统 
Subsystem /usr/local/sbin/sftpd



GNU/Linux SSH

配置 sshd:

1) 不允许 root 管理员登录

PermitRootLogin no

2) 不允许 root 在没有密钥的主机上登陆

PermitRootLogin without-password

3) 取消口令认证方式，仅使用密钥认证方式

PasswordAuthentication no

GNU/Linux SSH

sshd 配置文件概述：

4) 是否允许 root 管理员登录 [最好不]
PermitRootLogin yes

5) 设置是否允许空密码用户登录
PermitEmptyPasswords on

6) 设置是否使用口令认证方式
PasswordAuthentication yes



GNU/Linux SSH

sshd 启动：

```
#systemctl restart sshd.service
```

```
#systemctl start sshd.service
```

```
#systemctl stop sshd.service
```



GNU/Linux SSH

sshd 其他限制

1) 指定允许的用户 ssh

在 sshd_config 中最下行加入
AllowUsers 用户名

2) 指定用户不允许使用 ssh

DenyUsers 用户名

3) 指定某个账户来自指定的 IP 登录

AllowUsers 用户名 @ip_address



GNU/Linux SSH

sshd 其他限制

4) 指定允许 IP 连入 ssh

```
#vi /etc/hosts.allow  
sshd:192.168.0.1:allow
```

5) 拒绝所有 IP 链接

```
#vi /etc/hosts.deny  
sshd:ALL
```

4) 与 5) 两个文件都写入，则表示仅允许
192.168.0.1 允许链接 ssh



GNU/Linux SSH

SSH 的使用

1. 登陆远程主机

```
#ssh username@remote_hostname
```

例：

```
#ssh snow@192.168.1.123
```



GNU/Linux SSH

SSH 的使用

2. 登陆远程主机并启用远程主机图形程序

要求：两台主机均为 GUI 模式

```
#ssh -X username@remote_hostname
```

例：

```
#ssh -X snow@192.168.1.123
```



GNU/Linux SSH

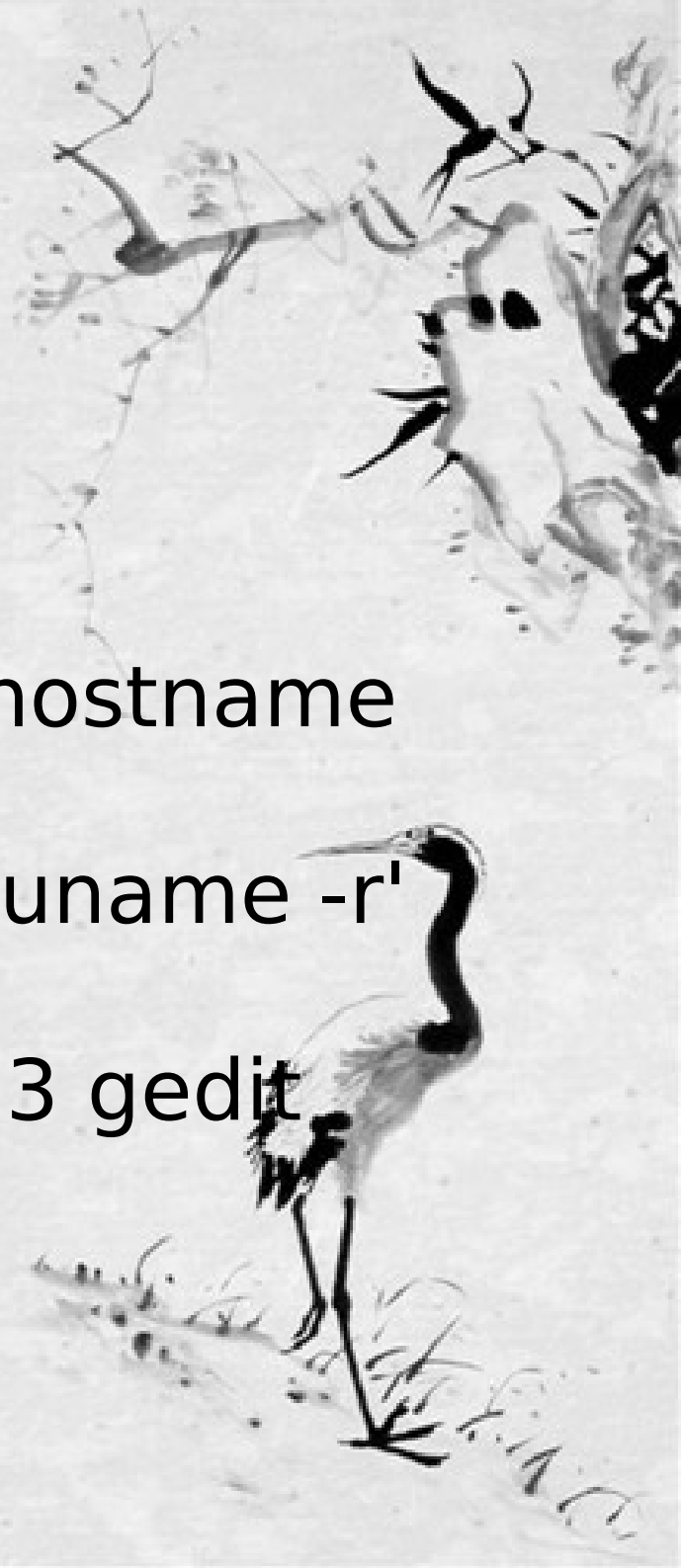
SSH 的使用

3. 执行远程主机 hostname 命令

```
#ssh snow@192.168.1.123 hostname
```

```
#ssh snow@192.168.1.123 'uname -r'
```

```
#ssh -X snow@192.168.1.123 gedit
```



GNU/Linux SSH

如 Linux 为 CLI 安装而并没有安装 X11-Server
则可以仅安装 xorg-x11-xauth

```
#yum install xorg-x11-xauth
```

并确认 sshd_config 的配置文件中以下配置为

```
AllowTcpForwarding yes
```

```
X11UseLocalhost yes
```

```
X11DisplayOffset 10
```

```
X11Forwarding yes
```

重新启动 SSH 服务，以测试 X11 转发

GNU/Linux SSH

如 Linux 已经安装了图形化，则 xauth 已经安装成功。

而且与 CLI 模式一样，不必要开启 X-Server. 如果有需求在 Server 端开启 X, 则可以执行

```
#X -br & /* 默认监听 6000/tcp
```

-br 为在以黑色背景下创建一个 X 根窗口

*** 在 ssh 服务器端不用必须打开 X Server**

GNU/Linux SSH

Windows 下使用 SSH 管理远程 Linux/Unix

注意及说明：

如果执行有问题请安装 xauth,xhost 软件包。

并

将 xhost 加入服务器的 IP 地址：

```
#xhost + X_window_server_ip
```



GNU/Linux SSH

Windows 下使用 SSH 管理远程 Linux/Unix

工具名称 :putty

1. 远程 CLI 管理

2. 远程启用 GUI 程序



GNU/Linux SSH

Windows 下使用 SSH 管理远程 Linux/Unix

1. putty 使用说明 (CLI)



GNU/Linux SSH

Windows 下使用 SSH 管理远程 Linux/Unix

2. putty 使用说明 (GLI)

1) 下载 windows 的 X-server 工具 xming

<http://sourceforge.net/projects/xming/>

2) 安装 xming

安装完成后将在 Windows 任务栏右下角出现一个“X”标记。将鼠标放置在标记处，将出现“Xming server:0.0”

GNU/Linux SSH

Windows 下使用 SSH 管理远程 Linux/Unix

2. putty 使用说明 (GLI)

3) 安装 xming 示意图

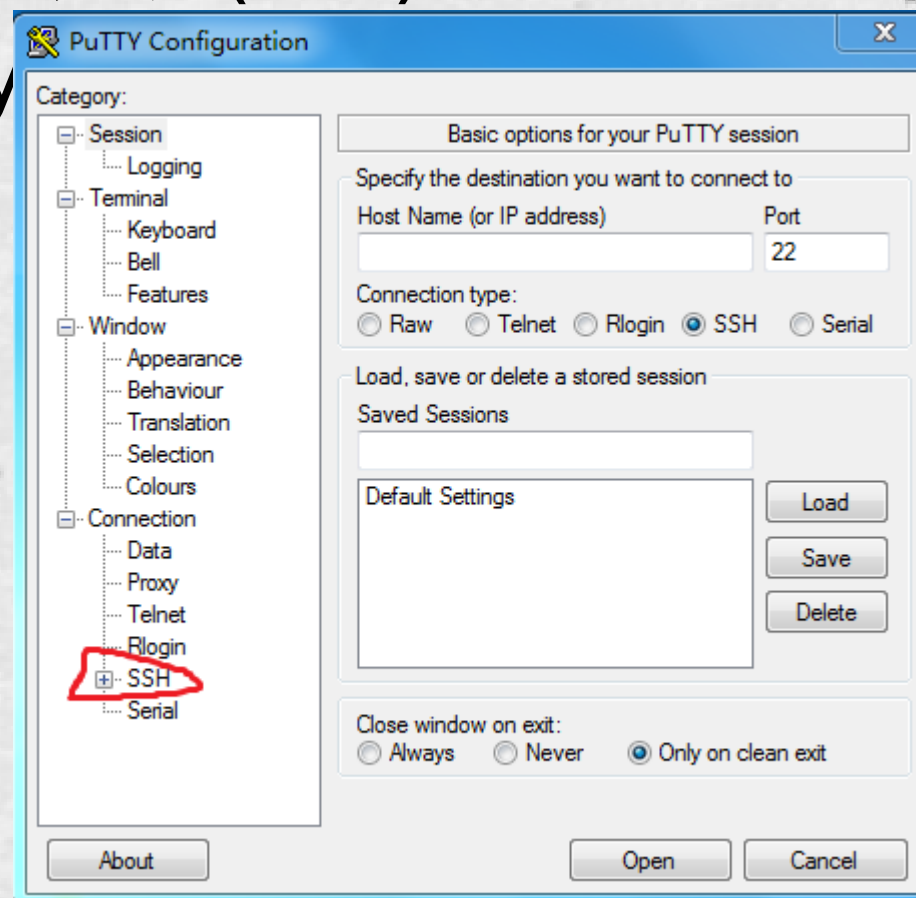


GNU/Linux SSH

Windows 下使用 SSH 管理远程 Linux/Unix

2. putty 使用说明 (GLI)

4) 设置 putty

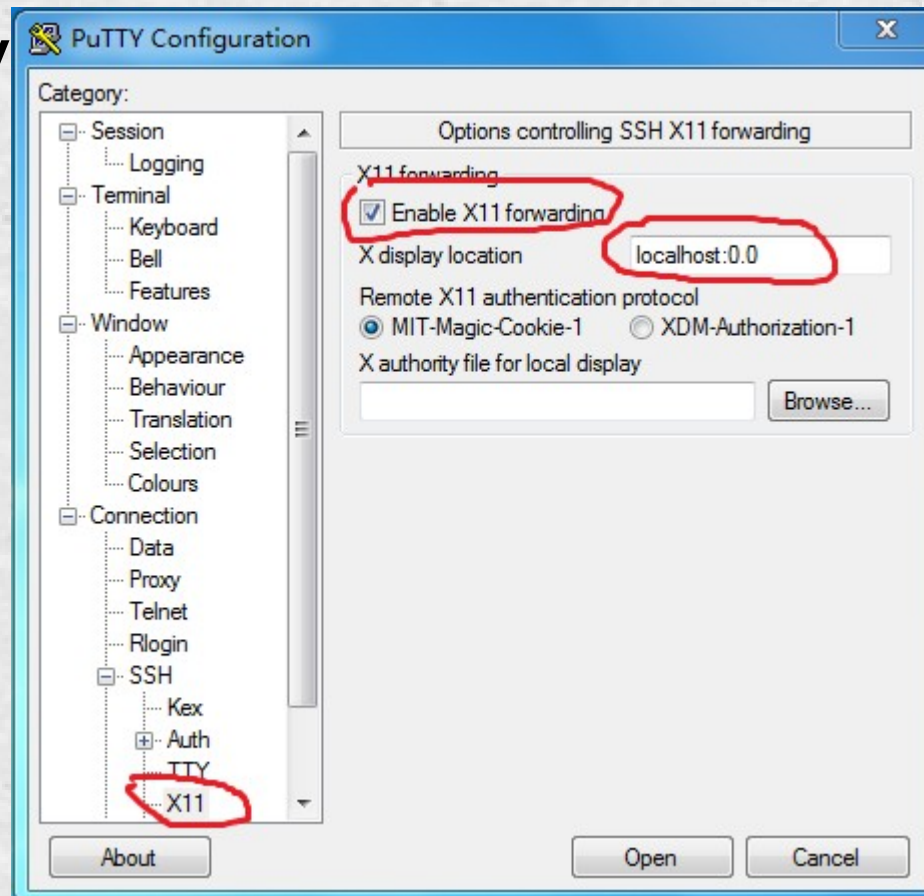


GNU/Linux SSH

Windows 下使用 SSH 管理远程 Linux/Unix

2. putty 使用说明 (GLI)

4) 设置 putty

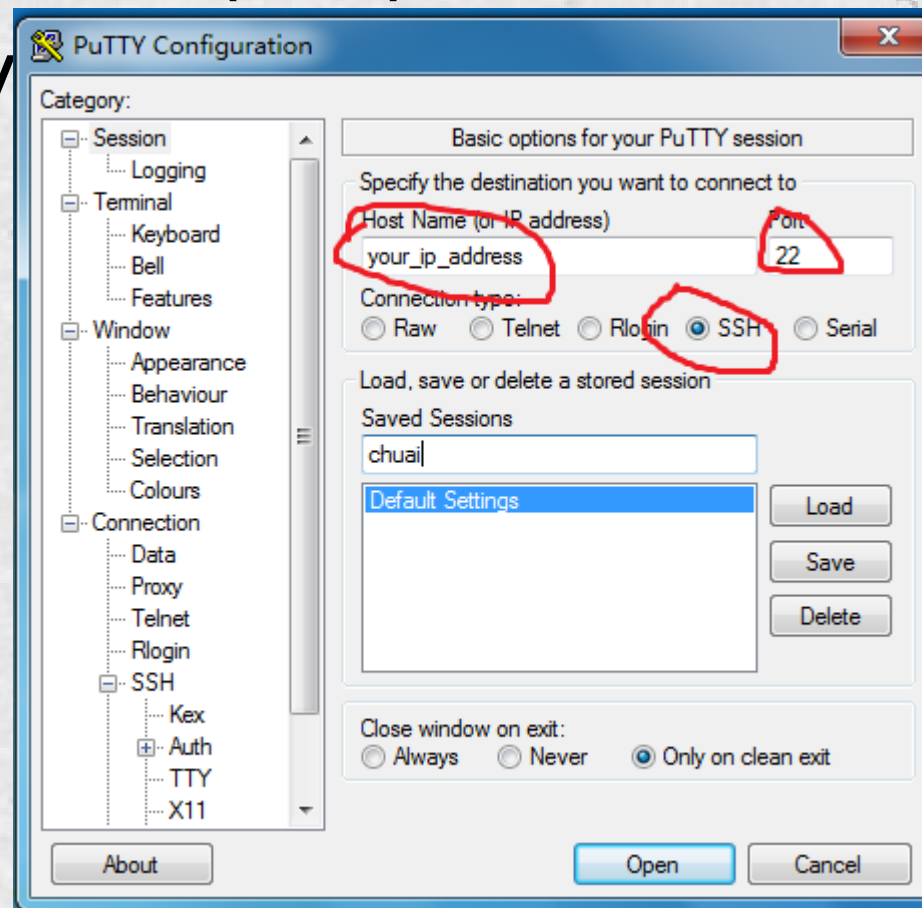


GNU/Linux SSH

Windows 下使用 SSH 管理远程 Linux/Unix

2. putty 使用说明 (GLI)

4) 设置 putty



GNU/Linux SSH

Windows 下使用 SSH 管理远程 Linux/Unix

2. putty 使用说明 (GLI)

5) 使用 putty 登陆 remote_host

6) 执行远程主机的 GUI 程序



GNU/Linux SSH

基础秘钥方式登陆远程主机

1. 在当前用户下生成秘钥

```
#ssh-keygen
```

(当提示输入口令时, 可以为空)

2. 查看秘钥

```
#less ~/.ssh/id_rsa ← 私钥
```

```
#less ~/.ssh/id_rsa.pub ← 公钥
```



GNU/Linux SSH

基础秘钥方式登陆远程主机

3. 将公钥复制至远程主机

```
#ssh-copy-id snow@192.168.1.123
```

4. 测试

```
#ssh snow@192.168.1.123
```

5. 登陆远程主机后，可查看刚刚复制的秘钥与公钥一致

```
$less ~/.ssh/authorized_keys
```



GNU/Linux SSH

多个公钥可以全部放在 `authorized_keys` 文件中，每个公钥一行。即可以完成多个设备之间相互认证。

```
[root@node1 .ssh]#scp id_rsa.pub  
root@node2:~/.ssh/id_rsa.pub.node1  
[root@node1 .ssh]#scp id_rsa.pub  
root@node3:~/.ssh/id_rsa.pub.node1
```

```
[root@node2 .ssh]#scp id_rsa.pub  
root@node1:~/.ssh/id_rsa.pub.node2  
[root@node2 .ssh]#scp id_rsa.pub root@node3  
:~/.ssh/id_rsa.pub.node2
```

```
[root@node3 .ssh]#scp id_rsa.pub  
root@node1:~/.ssh/id_rsa.pub.node3  
[root@node3 .ssh]#scp id_rsa.pub
```



GNU/Linux SSH

多个公钥可以全部放在 `authorized_keys` 文件中，每个公钥一行。即可以完成多个设备之间相互认证。

```
[root@node1 .ssh]#cat id_rsa.pub* >>  
authorized_keys
```

```
[root@node2 .ssh]#cat id_rsa.pub* >>  
authorized_keys
```

```
[root@node3 .ssh]#cat id_rsa.pub* >>  
authorized_keys
```



GNU/Linux SSH

加快 ssh 登录

修改 sshd_config 中如下行 

/* 是否取消使用 ~/.ssh/.rhosts 来做为认证
IgnoreRhosts yes


/*GSSAPI 的用户认证登陆的时候客户端需要对服务器端的 IP 地址进行反解析，如果服务器的 IP 地址没有配置 PTR 记录，则容易卡住并超时。

GSSAPIAuthentication no 

GNU/Linux SSH

加快 ssh 登录

修改 sshd_config 中如下行

/* 不使用 DNS 尝试解析 IP 地址 

UseDNS no



GNU/Linux SSH

sshd 端口转发：

SSH 端口转发分为两种，一种是本地端口转发，又称为本地 SSH 隧道。一直是远程端口转发。SSH 端口转发，还必须指定数据传送的目标主机，从而形成点对点的端口转发。

ssh 服务配置文件中，需要开启
GatewayPorts yes
AllowTcpForwarding yes



GNU/Linux SSH

sshd 端口转发：

示例：

```
#ssh -l snow -L
```

```
192.168.7.27:59000:192.168.30.68:80
```

```
192.168.6.19
```

参数 -L 后面总共有四个用冒号分割的值，分别是 '本地地址：本地端口：目标主机：目标主机端口'。这条命令的意思，就是指定 SSH 绑定本地 192.168.7.27 的 59000 端口，然后指定 192.168.6.19 将所有的数据，转发到目标主机 192.168.30.68 的 80 端口。

GNU/Linux SSH

sshd 端口转发：

举例：

有 A,B,C 3 台服务器，A,C 有公网 IP, B 是某 IDC 的服务器无公网 IP. A 通过 B 连接 C 的 80 端口 ($A \rightleftharpoons B \rightleftharpoons C$), 那么在 B 上执行如下命令即可：

```
$ ssh -CfNg -L 6300:127.0.0.1:80 userc@C  
$ ssh -CfNg -R 80:127.0.0.1:6300 usera@A
```

GNU/Linux SSH

sshd 端口转发：

ssh 参数：

C 表示压缩数据传输

f 表示后台运行，并推荐加上 -n 参数。

N 表示不执行脚本或命令

g 表示允许远程主机连接转发端口



GNU/Linux SSH

sshd 端口转发：

ssh 参数：

L 表示本地转发

R 表示远程转发

q 安静模式，忽略一切对话和错误提示。

T 不占用 shell 后台用户验证，这个选项很有用，
没有 shell 的不可登陆账号也能使用。



GNU/Linux SSH

sshd 端口转发：

ssh 参数：

n 将标准输入重定向至 /dev/null

socket 代理

```
#ssh -qTfnN -D port remotehost
```

注意：

（用证书验证就直接主机名，无证书须要加用户名和密码）



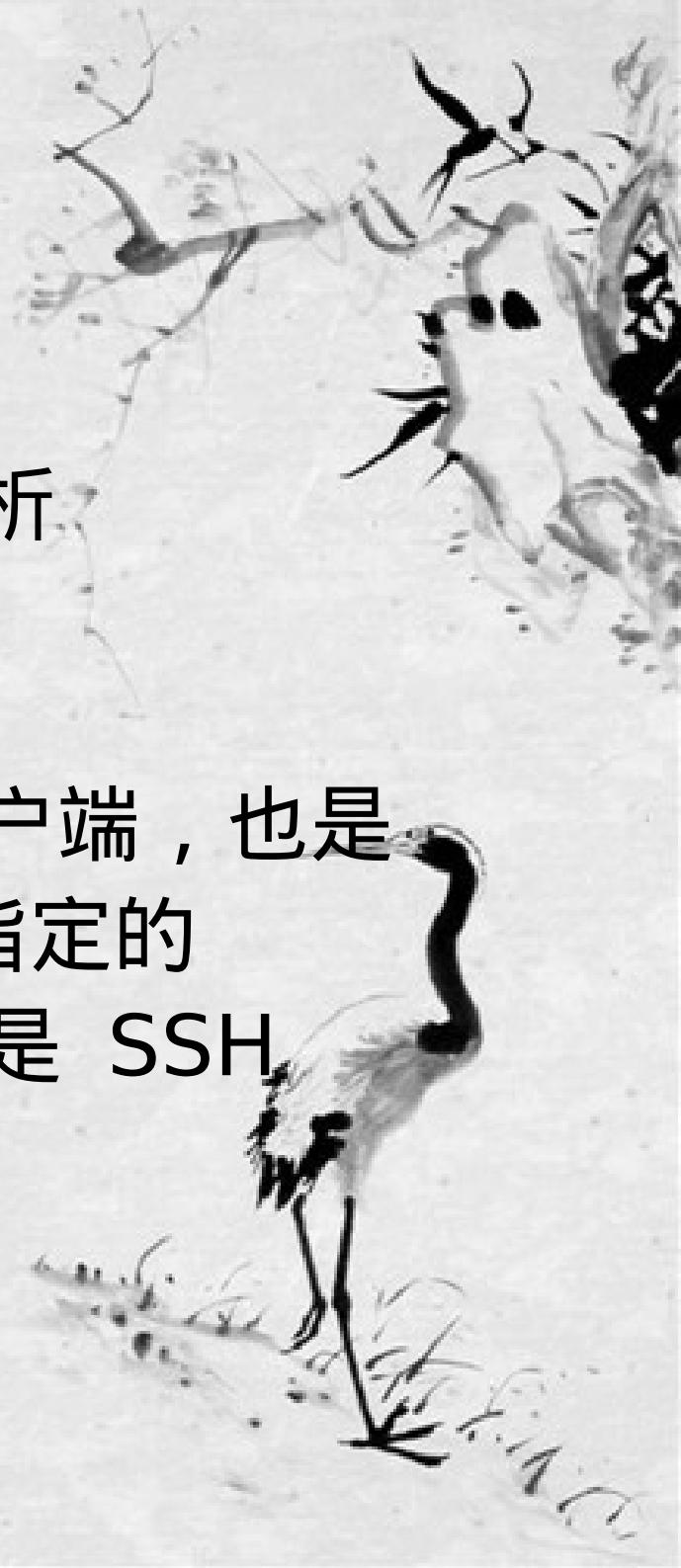
GNU/Linux SSH

sshd 端口转发：

本地转发与远程转发的对比与分析

本地转发时：

应用程序客户端同时是应用的客户端，也是 SSH Client, 这两个连接都链接到指定的 Server（既是 LDAP 服务端，也是 SSH Server）。



GNU/Linux SSH

sshd 端口转发：

本地转发与远程转发的对比与分析

远程转发时：

应用程序客户端是应用的客户端，但却是 SSH Server ；而应用程序服务器是应用服务端，但却是 SSH Client 。这样两个连接的方向刚好相反。

GNU/Linux SSH

ssh 子系统

1. 远程复制

命令 :scp

功能：将本地 / 远程文件复制到远程 / 本地



GNU/Linux SSH

ssh 子系统

语法格式：

#scp remote_account@remote_host
locale_path/filename

#scp locale_path/filename
remote_account@remote_host



GNU/Linux SSH

ssh 子系统

scp 选项：

1. 基本与 cp 一致
2. -C: 使能压缩选项
3. -P: 指定端口
4. -r: 递归，复制目录时使用



GNU/Linux SSH

ssh 子系统

scp 选项：

4. -4: 强行使用 IPV4 地址

5. -6: 强行使用 IPV6 地址

6. -v: 显示详细动作



GNU/Linux SSH

ssh 子系统

2. 远程文件传输

命令 :sftp

功能：将本地 / 远程文件上传或下载至远程 / 本地

语法格式：

```
#sftp remote_account@remote_host
```



GNU/Linux SSH

ssh 子系统之 sftp

将 sftp 账户 chroot

1) 创建一个组

```
#groupadd sftp_users
```

2) 将账户加入至组

```
#usermod -G sftp_users snow
```



GNU/Linux SSH

ssh 子系统之 sftp

3) 配置 ssh 服务文件

```
#cd /etc/ssh
```

```
#vi sshd_config
```

将 147 行注释

```
#Subsystem sftp
```

```
/usr/libexec/openssh/sftp-server
```

并改为以下内容

```
Subsystem sftp internal-sftp
```



GNU/Linux SSH

ssh 子系统之 sftp

3) 配置 ssh 服务文件

在最下行添加如下内容

```
Match Group sftp_users
```

```
  X11Forwarding no
```

```
  AllowTcpForwarding no
```

```
  ChrootDirectory /home
```

```
  ForceCommand internal-sftp
```



GNU/Linux SSH

ssh 子系统之 sftp

4) 重新启动 sshd 服务

5) 测试

