远程控制服务

配置sshd服务

sshd是基于SSH协议开发的一款远程管理服务程序，不仅使用起来方便快捷，而且能够提供两种安全验证的方法：

- 基于口令的验证—用账户和密码来验证登录；
- 基于密钥的验证—需要在本地生成密钥对，然后把密钥对中的公钥上传至服务器，并与服务器中的公钥进行比较；该方式相较来说更安全。

sshd服务的配置信息保存在`/etc/ssh/sshd_config`文件中。运维人员一般会把保存着最主要配置信息的文件称为主配置文件，而配置文件中有许多以井号开头的注释行，要想让这些配置参数生效，需要在修改参数后再去掉前面的井号。

sshd服务配置文件中包含的参数以及作用

参数	作用
Port 22	默认的sshd服务端口
ListenAddress 0.0.0.0	设定sshd服务器监听的IP地址
Protocol 2	SSH协议的版本号
HostKey /etc/ssh/ssh_host_key	SSH协议版本为1时，DES私钥存放的位置
HostKey /etc/ssh/ssh_host_rsa_key	SSH协议版本为2时，RSA私钥存放的位置
HostKey /etc/ssh/ssh_host_dsa_key	SSH协议版本为2时，DSA私钥存放的位置
PermitRootLogin yes	设定是否允许root管理员直接登录
StrictModes yes	当远程用户的私钥改变时直接拒绝连接
MaxAuthTries 6	最大密码尝试次数
MaxSessions 10	最大终端数
PasswordAuthentication yes	是否允许密码验证
PermitEmptyPasswords no	是否允许空密码登录（很不安全）

安全密钥验证

1. 在客户端主机中生成“密钥对”

```
[root@localhost ~]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
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:
SHA256:9+KE/GYBG6wjbCQ4o9j139nD9kkrL29bdAYd49kTvLo
root@localhost.localdomain
The key's randomart image is:
+---[RSA 2048]----+
|              .+ |
|              .o*|
| .     .      .++|
|+ . o   +      o.|
|o+ = . .S+.    . +|
|o . + +..o.. . o.|
|   . . oo.o=. o .|
|        .+=.BE.+ |
|         oo. OB. |
+----[SHA256]-----+
```

2. 把客户端主机中生成的公钥文件传送至远程主机

```
[root@localhost ~]# ssh-copy-id 192.168.91.128
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed:
"/root/.ssh/id_rsa.pub"
The authenticity of host '192.168.91.128 (192.168.91.128)' can't be
established.
ECDSA key fingerprint is SHA256:PWPGI+gebAxdFtOfQe66cO/RnTBEV/Qw5AEoZv6w5lM.
ECDSA key fingerprint is
MD5:61:3d:ae:39:43:65:70:f4:9a:10:ff:48:67:6f:ef:54.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to
filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are
prompted now it is to install the new keys
root@192.168.91.128's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh '192.168.91.128'"
and check to make sure that only the key(s) you wanted were added.
```

3. 对服务器进行设置，使其只允许密钥验证，拒绝传统的口令验证方式。记得在修改配置文件后保存并重启sshd服务程序

```
[root@localhost ~]# vim /etc/ssh/sshd_config
...................
65 PasswordAuthentication no
...................
[root@localhost ~]# systemctl restart sshd
```

4. 在客户端尝试登录到服务器，此时无须输入密码也可成功登录

```
[root@localhost ~]# ssh 192.168.91.128
Last login: Fri Apr 19 17:12:37 2019 from 192.168.91.1
```

不间断会话服务

当与远程主机的会话被关闭时，在远程主机上运行的命令也随之被中断。

screen是一款能够实现多窗口远程控制的开源服务程序，简单来说就是为了解决网络异常中断或为了同时控制多个远程终端窗口而设计的程序。用户还可以使用screen服务程序同时在多个远程会话中自由切换，能够做到实现如下功能。

- 会话恢复：即便网络中断，也可让会话随时恢复，确保用户不会失去对远程会话的控制。
- 多窗口：每个会话都是独立运行的，拥有各自独立的输入输出终端窗口，终端窗口内显示过的信息也将被分开隔离保存，以便下次使用时依然能看到之前的操作记录。
- 会话共享：当多个用户同时登录到远程服务器时，便可以使用会话共享功能让用户之间的输入输出信息共享。

```
[root@localhost ~]# yum -y install screen
```

管理远程会话

screen命令能做的事情

- 用`-S`参数创建会话窗口
- 用`-d`参数将指定会话进行离线处理
- 用`-x`参数一次性恢复所有的会话
- 用`-ls`参数显示当前已有的会话
- 用`-wipe`参数把目前无法使用的会话删除

```
[root@localhost ~]# screen -S window
```

虽然看起来与刚才没有不同，但实际上可以查看到当前的会话正在工作中

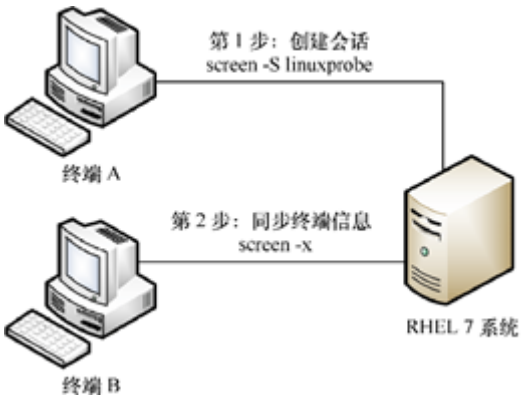
```
[root@localhost ~]# screen -ls
There is a screen on:
        7758.window     (Attached)
1 Socket in /var/run/screen/S-root.
```

要想退出一个会话也十分简单，只需在命令行中执行exit命令即可

在日常的生产环境中，其实并不是必须先创建会话，然后再开始工作。可以直接使用screen命令执行要运行的命令，这样在命令中的一切操作也都会被记录下来，当命令执行结束后screen会话也会自动结束

```
[root@localhost ~]# screen ping -c 4 baidu.com
[screen is terminating]
[root@localhost ~]#
```

会话共享功能



堡垒机

概述

当今的时代是一个信息化社会，信息系统已成为各企事业单位业务运营的基础,由于**信息系统运维**人员掌握着信息系统的最高权限，一旦运维操作出现安全问题将会给[企业]或单位带来巨大的损失。因此,加强对运维人员操作行为的监管与审计是[信息安全]发展的必然趋势。在此背景之下,针对运维操作管理与审计的**堡垒机**应运而生。堡垒机提供了一套多维度的运维操作控管控与审计解决方案，使得管理人员可以全面对各种资源(如[网络]设备、[服务器]、安全设备和[数据库]等)进行集中账号管理、细粒度的权限管理和访问审计，帮助企业提升内部风险控制水平。

堡垒机，即在一个特定的网络环境下，为了保障网络和数据不受来自外部和内部用户的入侵和破坏，而运用各种技术手段监控和记录运维人员对网络内的服务器、网络设备、安全设备、数据库等设备的操作行为，以便集中报警、及时处理及审计定责。

目前国内外有很多厂商生产堡垒机，堡垒机目前有两种最主流的形式。

硬件堡垒机：厂商定制硬件设备，并且将软件系统封装在其中，比较封闭，安全系数较高

软件堡垒机：厂商仅仅维护软件堡垒机系统，需要客户自己部署堡垒机，可以虚拟化形式，部署自由程度高

核心功能

- 登录功能
- 账号管理
- 身份认证
- 资源授权
- 访问控制
- 操作审计

JumperServer

- JumpServer 是全球首款完全开源的堡垒机, 使用 GNU GPL v2.0 开源协议, 是符合 4A 的专业运维审计系统。
- JumpServer 使用 Python / Django 进行开发, 遵循 Web 2.0 规范, 配备了业界领先的 Web Terminal 解决方案, 交互界面美观、用户体验好。
- JumpServer 采纳分布式架构, 支持多机房跨区域部署, 中心节点提供 API, 各机房部署登录节点, 可横向扩展、无并发访问限制。
- 官方文档：https://docs.jumpserver.org/zh/master/