15.6.5 Configure OpenSSH

Click one of the buttons to take you to that part of the video.

Configure OpenSSH 0:00-0:37

In this demonstration we're going to review how to establish an encrypted connection between two computers over the network, using SSH. Usually when you install a Linux distribution, the SSH package gets installed by default.

Let's verify that that's the case on this system. Oops, I typed my password wrong. Start again. Let's run the 'rpm -qi' command and the name of the SSH package is 'openssh'. It has indeed been installed on this distribution.

SSH Daemon 0:38-4:02

Understand that there are two different pieces to an ssh implementation. We have the SSH server, which is the system we're going to connect to; and we have the SSH client, which is the system we connect from.

Let's use this system as our SSH server. This is where our sshd daemon will need to run. We'll connect to it from a different system, using the SSH client.

The SSH daemon is configured using a file in the '/etc/ssh' directory. I'll run the 'ls' command. We see that there are several different files in this directory. Notice that there are two, however, that are config files: sshd_config and ssh_config.

It's very important that you don't confuse these two files. sshd_config is used to configure the SSH daemon, while ssh_config is used to configure the SSH client. If you're configuring the system, make sure that you're editing the right file.

In this case, we're going to be running the SSH daemon on this system. We want to edit the 'sshd_config' file. In all actuality, we really don't have to do anything, because usually you don't have to actually change a whole lot in this configuration file.

It's designed to work right out of the box. There are a few parameters that you could configure if you wanted to. If you just want to get everything up and running, you actually don't have to do anything. I'm actually going to get out of this file without making any changes or saving any changes.

The next thing we need to do is see whether or not the SSH daemon is running on this system. We'll enter 'systemctl status sshd', and it tells us that the service is there and available, but it is currently inactive.

What we need to do is go ahead and start it. We'll run 'systemctl start sshd' and now let's run the 'status' command again. We see that it is now active, it's up and running, and we see that it is actively listening on tcp port 22.

This right here 0.0.0.0 tells us that it is listening on all configured IPv4 addresses on this system. There's only one. It really doesn't matter. It's also listening over IPv6 as well on the same port.

Also be aware that before we can make a connection to this system, we have to open up port 22 in this host firewall, otherwise you won't be able to connect.

The way you do that will vary between distributions. On this one, I believe I have a firewalld configuration app installed right here. If I scroll down here and click on launch, I do have to authenticate as root before I can manage the firewall.

Over here under services I want to scroll down and make sure that ssh is marked, and it is, meaning that all SSH traffic will be allowed through our host-based firewall on this system.

I can't tell you how many times I've tried to troubleshoot an SSH connection that won't establish, only to discover that the firewall was running and port 22 was blocked.

At this point, let's go ahead and switch over to a client system and let's just use the SSH client to connect to this system over the network.

SSH Client 4:03-7:46

To establish a basic SSH connection using the SSH client, I enter 'ssh' at the shell prompt, followed by an '-l', and then the name of the user on the other system that I want to connect as.

This is a point of confusion for folks when they first start using SSH. You've got to remember that there are two sets of user accounts involved here. I've got the user accounts here on my client system, and I've got a separate set of user accounts on my server system.

When we log in to the remote system using the SSH client software, we have to authenticate as a user on the remote system.

In this case, we're going to log in to the remote system as the rtracy user. Notice I'm logged in to the local system as the ksanders user. I specify 'ssh -l rtracy', and then I have to specify the IP address or the hostname of the system that we want to connect to.

The system that we just configured SSH daemon on is fs5. I'll enter 'fs5.corpnet.com'. You could use an IP address instead if you'd rather. I'll press Enter.

Because this is the first time that I've connected to that other system fs5 using SSH, I'm prompted to accept the security key from that system that will be used to encrypt the data that gets passed back and forth between the client and the server. Enter 'yes' to accept it. The key is added to the list of known hosts.

Now I have to specify rtracy's password on the remote system, and I have logged in. Notice that the prompt down here has changed. Notice that I was ksanders@openSUSE. Now I'm rtracy@fs5.

At this point, I am working at the prompt of the fs5 system as if I were sitting in front of it. Any of the commands that I run at the prompt now here will be run on the fs5 system instead of my local system. For example, if I were to run the 'ls' command, it is run on the remote system. I see all of the files located in the rtracy user's /home directory on that other system.

Let's go ahead and close the connection between the client and the server. Enter 'exit' to do that. We are logged out and the connection is closed.

Now let's do something a little bit different with SSH. What we're going to do here is establish a new SSH connection between the client and the server on the remote system, run a command, and then immediately log back out.

This can be really useful in situations where you just need to run a single command on the remote system and you don't want to spend a whole lot of time there.

To do this, I enter 'ssh -l' again and we'll again connect as the rtracy user, just like we did last time. Then we specify the name of the host that we want to connect to--just like we did last time. In fact, I'm just going to press the up arrow key so I don't have to type at all.

Then we specify the command that we want to run on the remote system. In this example, let's do what we did last time and just run the ls command. In fact, let's change it a little bit. Let's do the 'ls -l' command. This is will generate a listing, a long format listing, of all the files in the rtracy user's /home directory on the fs5 system.

Press Enter. Notice that something is different this time. We weren't prompted to accept the key. We only have to do that once. Now that we've accepted it, it's already there and all we have to do is type the password for the user on the other system.

Notice what happened. As soon as I connected and authenticated, the 'ls -l' command that we specified right here was run. The output is displayed locally here on this system and then the connection was closed. I'm back to ksanders@openSUSE.

'scp' Command 7:47-11:39

In addition to the SSH client, the openSSH rpm package also provides other security utilities that you can use to move data back and forth between these two systems. For example, it includes a utility called scp. scp stands for secure copy. It allows us to either copy a file up to, or copy a file down from, the remote system securely using SSH encryption.

For example, let's suppose I want to copy a file from the rtracy user's /home directory on the remote system. I want to copy it down to the ksander's directory here on this system. Let's copy this c programming file right here--zombie.c.

To do this, I would enter 'scp' and then the user that I want to connect to the remote system as. We'll do 'rtracy' again. Then we enter an '@' sign and then we enter the hostname or the IP address of the remote system where the SSH daemon is running.

We'll enter again 'fs5.corpnet.com'. Then we add a space and then we enter the name of the file on the remote system that we want to copy down.

Because we're going to be placed initially in the rtracy user's /home directory by default on the remote system when we authenticate, we don't actually have to specify the full path to the file that we want to copy--although we could if we wanted to. It wouldn't hurt anything.

We'll just specify 'zombie.c' in this case. Then we have to specify where we want that file copied to here on this system. Let's just put it in '~' --our home directory. Press Enter.

You know what? I just realized what I did wrong. I made a typo here right between the hostname and the filename; we have to put a colon ':' here.

This specifies we're going to copy as rtracy at this hostname, the zombie.c file, and we're going to drop it into our user's /home directory. Press Enter. I have to enter the rtracy user's password on the remote system, and the file is copied down.

We do the 'ls' command now, and we see the zombie.c file that we copied from the rtracy user's /home directory on fs5 to the ksanders /home directory on this system. It can work in the other direction as well.

We can copy files up from this computer to the remote computer. Use 'scp' again, and this time the syntax is going to be a little bit different because we're going the other direction. The first thing we have to specify in this scenario is the name of the file that we want to copy up to fs5.

For example, let's copy this file right here, Project_design.odt. We specify the 'Project_design.odt' file located in my ksanders user's /home directory on this system. Then we have to specify the username that we want to connect to the remote system as.

As before, we'll connect as the 'rtracy' user '@' and then the IP address or hostname of the remote system where the SSH daemon is running, 'fs5.corpnet.com' and then a ':' at the end. That tells it to put the file in the /home directory of the rtracy user. Press Enter. I enter rtracy's password on the remote system, and the file is copied up.

Let's use the same command that we ran earlier to execute ls on the remote system to view a listing of files in the rtracy's home directory on fs5. There's the file that we copied up, Project_design.odt.

'sftp' Command 11:40-13:31

The last SSH utility that I want to show you is sftp. The sftp command can be used to transfer files between two systems, just like you would if you were using the ftp command. The key difference, though, is that sftp does so securely.

ftp is notoriously insecure. Usernames and passwords are sent via ftp cleartext, and the data that's transferred via ftp is sent cleartext. With sftp, we encrypt all that information so it can't be sniffed.

The syntax is 'sftp' followed by the username on the remote system that we want to authenticate as 'rtracy@', and then the IP address or hostname of the remote system where the SSH daemon is running. 'fs5.netcorp.com' again. Hit Enter. You have to enter the password for rtracy on the remote system. And we're connected to fs5, and the fstp prompt is displayed.

At this point, I can use most of the commands I would use if I'd use the ftp utility to connect. For example, I can enter 'pwd' to see what the current directory is on the remote system. You can see that the remote working directory is /home/rtracy.

We can also run the 'ls' command to generate a listing of files on the remote system. We can also use the 'get' command to download a file from the remote system to my home directory here.

Let's go ahead and get the zombie file this time. We got the zombie.c file previously. This time we'll get the compiled version of that file called zombie. 'get zombie' and it's copied down to my /home directory here on this system. Enter 'exit', get out of sftp. I'll run the 'ls' command and here's the file that we just downloaded.

Summary 13:32-13:44

That's it for this demonstration. In this demo we learned about how you can use SSH to establish secure connections between two Linux systems. We first looked at the sshd daemon. We then looked at the SSH client. Then we looked at other SSH utilities, such as scp and sftp.
