How To: Enable SSH

In today’s article, we’ll learn about enabling SSH. SSH is a useful tool for remotely managing your Linux computer. This is a pretty simple, painless, and quick exercise.

Let’s say your computer is in another room, another state, another country, or on another continent entirely. How are you going to manage it? After all, we have servers across the world and it is not even remotely economical to send a person to administer each one of them in person.

On a home-use note, it’s perfectly suitable to manage my own router using SSH. It’s quick, easy, lightweight, effective, doesn’t require an attached monitor, and more. What’s not to like?

Installing SSH on Linux

My homemade router doesn’t have a keyboard attached. It doesn’t have a monitor attached, so it’s not like I can just easily walk over and deal with it.

I just got a new computer, a lovely laptop that I got for a fantastic price. I got it to test Lubuntu. I don’t always want to have to go over to the device and physically use it.

I have a dedicated server in Las Vegas. I live in Maine. It wouldn’t be practical to fly out to Vegas every few days to run updates on the server. It wouldn’t make financial sense to go out there every time the server needed to be rebooted.

These are all great candidates for SSH. SSH stands for “Secure Shell” and it’s a protocol. It’s not an application, and you can use SSH for all those things. A great many applications can communicate over the SSH protocol, including every terminal emulator that I’m familiar with.

The man page defines SSH as:

> *ssh – OpenSSH remote login client*

It has been around since the mid-90s and does nifty things beyond allowing you to control the remote system with commands, it also allows you to transfer files with things like SCP and SFTP. It's right full of nifty features and you might as well become familiar with it. I'll possibly cover both of those in a future article.

It's very, very easy to get going. Simple use your keyboard to open the terminal, by pressing CTRL + ALT + T.

Now, simply type (this one can be easily adjusted to other package managers):

[code]sudo apt install openssh-server[/code]

There… You're done. Well, you are more or less done. You now effectively have SSH running. It starts itself after you install it. That's pretty handy!

Now, make sure you're on the same subnet, and you can connect to the remote computer – the computer where you just installed and enabled SSH. You can do that in a couple of ways. You can do it like this:

[code]ssh remote_user@ip.a.d.d.r[/code]

You can switch the obvious for the obvious, but you will need to know the IP address for the remote computer. That seems a bit tedious, so let's just skip that part. Rather than remembering the IP address (which may change), just remember the name of the device.

So, instead, you'd run:

[code]ssh remote_user@host_name.local[/code]

If that doesn’t make sense, this is how I’d connect to the new laptop:

[code]ssh kgiii@kgiii-msi.local[/code]

Obviously, the hostname is ‘kgiii-msi’ and your hostname will be different. It’ll be the name you gave the computer during the installation process, typically during the same phase where you generated your user account. If you don’t actually know your hostname, you can easily find it. It’s simply:

[code]hostname[/code]

There’s more to this and we’ll likely cover that soon enough, but that’ll get you started. If you have a firewall installed and enabled, you may need to let SSH through if you want to use it. This is such a simple thing that I’d be remiss in my duties to not make folks aware of how simple it is.

And, with that, I thank you my dear reader for taking the time out of your day to humor this old fool. Your feedback is appreciated and keep signing up to that whole newsletter thing. Being old, I tend to forget to submit and share these articles elsewhere. Signing up means you have no excuses for missing an article!