Install SSH to Remotely Control Your Linux Computers

If this article's headline looks vaguely familiar, it's probably because I previously wrote an article that told you how to install SSH. That article was on the earlier site. This article will show you how to install SSH, so that you can remotely control your Linux computers.

The old site, while up, will redirect to this address. It's also a bit of a misnomer. We'll be installing 'OpenSSH' and enabling SSH. SSH is the protocol, OpenSSH is the application, specifically the 'openssh-server'.

Let's say your computer is in another room, in another state, over in another country, or perhaps on another continent entirely. How are you going to manage it? Servers are scattered across the world and it is not even remotely economical to send a person to administer each one of them in person. You'll need to manage these devices remotely.

One of the best tools for this job is SSH.

As a home-use note; SSH is perfectly suitable to manage my own router. It's quick, easy, lightweight, effective, doesn't require an attached monitor, and more. What's not to like? I SSH into my computers all the time! In fact, right this minute I'm connected to two other computers via SSH!

SSH has been around since 1995 and it lets you issue commands on a remote computer. In fact, the man page describes it like this:

> *ssh — OpenSSH remote login client*

Which, as you can guess, means it lets you login to remote computers so that you can control them. It's a pretty handy

tool to have in your toolbox and it’s actually simple to install.

Install SSH:

SSH is really the protocol, and you can do many things over it. OpenSSH is the application that we’ll really be installing. Once that’s installed, you can login to the computer remotely and manage it that way. I use it quite often.

My homemade router doesn’t have a keyboard attached. It doesn’t even have a monitor attached. It’s not like I can just easily walk over and deal with it. I just got a laptop that, and it’s only used to test Lubuntu. I don’t always want to have to go over to the device and physically use it to start the test.

There’s also my dedicated server in Las Vegas – and 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!

Those are all situations where you can use SSH. It’s available in pretty much every default repository out there. I’m surprised more people don’t use it. To get started, you just open your terminal (press CTRL + ALT + T on your keyboard) and enter the following:

If you’re using a distro with apt:

[crayon-60d28ffe645dc191654479/]

If you’re using a distro with yum:

[crayon-60d28ffe645e7295576985/]

Simply adjust that for your distro. For example, in OpenSUSE you may have installed it during the OS installation process.

If it wasn't installed during the initial OS installation then it's just called 'openssh' – if you want to install a few of the applications surrounding openssh. You can also do a 'sudo zypper install openssh-server' like the rest of us.

Anyhow, once you've installed it, you may not even need to start it. If you install it on Ubuntu, you can go right ahead and test it immediately. If your distro requires that you start it manually, you should do that.

Once you're done, you can test it easily enough. Try this:

[crayon-60d28ffe645e9861937985/]

If that works, you've properly installed openssh-server and can now make use of SSH. You may also need to enable it in your firewall. Chances are that your firewall knows what SSH is.

To connect to your device from a remote computer, you'd do:

[crayon-60d28ffe645eb468423596/]

You can use a specific username in that command, like demonstrated in the testing command just above this command. It's not mandatory, but doing so will skip a step.

Closure:

You can expect a couple more SSH articles, as *this is woefully incomplete*. A lot more can be done with SSH, plus SSH should be properly secured. For most of you, behind a NATed router, you don't really have to worry too much unless you enable port forwarding. If you're making the port available to the world-wide-web, you're definitely going to need to add some security. Otherwise, there are a few nifty things you can do once SSH is enabled. We'll cover those in future articles.

You can also connect with your hostname – probably. In many instances, you'd do this (distro-dependent):

[crayon-60d28ffe645ed790413169/]

For example, to connect to my testing laptop, I use the following command:

[crayon-60d28ffe645ef570759231/]

Go ahead and play around with it. There's a number of ways to help secure SSH and we'll go over some of that in a future article. I've been maintaining the 'article every two days' thing for a while now. I see no reason to expect that to not continue.

Y'all have been chewing through bandwidth, so if you'd like to donate that'd be awesome. There are also ads that you can unblock. It's very much against Google's rules to ask people to click on them, but I can ask you to unblock them!

You can also register to help, write an article of your own, leave comments, vote on articles, sign up for the newsletter below, or even help support the site by buying inexpensive hosting. Equally important, please share these articles on social media! The more traffic, the happier I am – even if I've gotta pay for it myself! Until next time…