# How to setup SSH certificate for Unix/Linux systems

## Background:

SSH creates a secure communication channel between two servers. We call these two servers a client server and a remote server. The client initiates the connection and the server accepts the connection. Instead of requiring a password to login, SSH authenticates both servers using a public and private key. The public key, stored in the server, is used to encrypt messages that are sent to the client and the private key, stored in the client, is used to decrypt messages from the server. In the following steps, you will need access to a filesystem utility with a command line such as “terminal” or “command prompt” on both your client and server. Before setting up your connection, make sure that SSH is installed and running on both your servers. If you aren’t sure if SSH is installed, you can follow the steps in the link to verify and install it: How to setup an SSH server. Here is how to setup an SSH connection between two servers.

## Steps:

1. On your client system, open the filesystem utility (ex., terminal, command prompt, etc…)
2. On the command line, enter **ssh-keygen** to begin generating your public and private keys.
3. Follow the prompts to setup your keys.
   1. When prompted, enter the filename of your keys.
      1. To use the default filename, “id\_rsa,” simply press enter. It will store the public key “id\_rsa.pub” and the private key “id\_rsa” in the .ssh folder of your home directory.
      2. To use your own filename, for example “mykey,” write the full path name, **/home/{your username}/.ssh/mykey**. It will store the public key “mykey.pub” and the private key “mykey” in the .ssh folder of your home directory.
   2. If you wish to have a passphrase to protect your private key, enter a passphrase when prompted. You will be prompted twice to confirm your passphrase.
4. Enter **eval $(ssh-agent)** to load the ssh agent.
5. Add the private key to the ssh-agent.
   1. If you used the default filename “id\_rsa,” enter **ssh-add**.
   2. If you used your own filename, such as “mykey” enter **ssh-add $HOME/.ssh/mykey**.
   3. If you created a passphrase for the private key, enter it when prompted.
6. Add the public key to your remote server.
   1. Either use copy and paste utility
      1. Enter **cat id\_rsa.pub** to see the contents of the public key. (Use cat {filename}.pub if you chose a different name).
      2. Select the contents of the public key with your cursor and select “ctrl-c” or “command-c” to copy it.
      3. Login to your remote server and open the filesystem navigation utility.
      4. Enter **vi .ssh/authorized\_keys**.
      5. Go one line below the last line and right-click to paste the public key. To go down using Vi, select “shift-G > o > enter” on your keyboard or scroll down using “j”.
   2. Or use yank and paste from vi
      1. Enter **vi id\_rsa.pub** to open the public key file
      2. Enter “yy” to yank the public key text
      3. Exit vi used “esc > : > q”
      4. Login to your remote server and open the filesystem navigation utility.
      5. Enter **vi .ssh/authorized\_keys**.
      6. Go one line below the last line and enter “p” to paste the public key. To go down using Vi, select “shift-G > o > enter” on your keyboard or scroll down using “j”.
   3. Save and exit the authorized\_keys file by selecting “esc > : > wq” on your keyboard.
7. Set the permissions on the remote server to allow access to the public key.
   1. Enter **chmod 700 ~/.ssh**.
   2. Enter **chmod 644 ~/.ssh/authorized\_keys**.
   3. Enter **chmod 755 ~**.

Explanation of work:

For this how-to article, I chose to focus on teaching a relatively new learner to setup a basic ssh connection on their Unix/Linux system. As such, I avoided extraneous steps and provided the most essential information to complete the task. I included a background section explaining how SSH works and the purpose of using SSH versus password based authentication. I also provided a link on preliminary steps to download SSH (I have not actually linked anything, but I would ideally create a separate procedure explaining how to do this by verifying if SSH is installed by checking sshd, using yum/dnf to download the openssh server, test the sshd daemon, and open port 22 on the firewall for SSH connection). I spent my time performing a basic ssh setup and getting down a rough draft of the steps, then breaking steps into subsections, and finally adding additional formatting. I further edited text for accuracy and readability.

When writing, I made sure the content was readable and accessible. To make it readable, I made sure to use the same terminology for a concept (ex., always client for local server, filesystem utility for terminal/command prompt). I also visually made it easy to navigate. I used a heading 1 font for the title and a heading 2 font for subsections (background and steps) to create hierarchy. I also used the same format for all command line text with gray background and courier font to distinguish it from regular text and I used double quotes for keyboard commands that were not command line text (ctrl-c, yy, p, esc<:<wq).

While I tried to keep steps short and straightforward with active voice, I added an additional sentence if I thought it was necessary to explain the step. I chose to do this on the same line instead of adding the additional information in a sub-bullet because the reader will need to take extra time to shift their focus to the sub-bullet points. I only wrote subsections if there were extensive steps or conditions that needed elaboration (like in steps 3a, 5, and 7). In some instances I did not use active voice as the condition for executing the step was more important to state first than the action to be taken. For example, I chose to say “if you created a passphrase for the private key, enter it when prompted” as opposed to saying “enter the passphrase for the private key if you created one” because with the latter sentence, when they read “enter the passphrase..,” the reader is already preparing to do that before they realize they may not have created one, so it creates a lot of unnecessary mental backtracking.

I also wrote the steps with accessibility in mind by avoiding potentially ableist language. Some people who use technology cannot use their hands or vision, so using words like “click,” “type,” or “view” can leave out certain demographics. I also tried to make procedures screen reader friendly by avoiding spatial references like “above” or “below” or “to the left”.