## How to Install and Configure SSH on Ubuntu 22.04

The primary step in setting up remote access to your Ubuntu machine is to enable SSH. SSH, which stands for Secure Shell, is a network protocol that enables users to securely connect to remote resources over an unprotected network. This tool is indispensable for system administrators, developers, and anyone who needs to manage remote connections and transfer files safely. By leveraging SSH, you can confidently connect to your Ubuntu machine from another device, as all communications are secured through encryption, safeguarding against data theft and lessening the risk of remote network attacks.

Consider that you have obtained a cloud server from Hostman. To facilitate connection and administration of the server, SSH must be installed and configured. This guide will walk you through the installation and configuration process of SSH on Ubuntu 22.04.

## Requirements

Before you dive into installing and configuring SSH on your Ubuntu 22.04 device, make sure you have met a few essential prerequisites. Here’s a checklist of what you need:

**1. Introductory Awareness of the Linux Command Line Environment.**

Given that the process of installing and configuring SSH on Ubuntu 22.04 relies on terminal commands, therefore, it is essential to have a good understanding of fundamental Linux commands, including sudo, apt, nano, and systemctl. Additionally, one should be adept at navigating the command line, editing files, and overseeing services effectively.

**2. Root Access or Sudo Privileges**

Administrative (root) privileges are essential for installing software and configuring system services. The sudo command facilitates temporary access to these privileges. Consequently, one must either have a user account with sudo privileges or be logged in as the root user.

**3. Internet Connection**

The setup of the OpenSSH server, along with any supplementary packages, necessitates a functioning internet connection to retrieve the required files from repositories. It is essential to confirm that your Ubuntu machine is connected to the internet, whether through Ethernet or Wi-Fi.

**4. Firewall Configuration**

If your system has an active firewall, such as ufw, it might block SSH connections by default. To connect to your machine remotely, authorization of SSH traffic through the firewall will be required. Confirm whether ufw or any other firewall software is currently enabled, and if it is, ensure that it is configured to allow SSH access.

**5. Connecting to the Ubuntu Machine, Whether You're There in Person or Accessing It Remotely.**

It is essential to have physical access to the machine, or if you are configuring it remotely, the machine must be connected to the network to facilitate an SSH connection. When establishing SSH for remote access, make certain that the Ubuntu machine is linked to the network and can be accessed through its IP address.

**6. Accessing the OpenSSH Package.**

It is necessary for the openssh-server package to be accessible within the default Ubuntu repositories. Although this is usually the situation, it is essential to verify that your system's package list is current. To update the Ubuntu repositories, execute the command sudo apt update prior to installing OpenSSH.

**7. Accessing an Alternative Machine for SSH Testing**

Once SSH has been installed, it is advisable to verify the connection from an alternative machine, which may be another Ubuntu system or any other operating system that is compatible with SSH. To perform this test, you will require an additional device, utilizing either the terminal or an SSH client such as PuTTY on Windows.

**8. SSH Key Pair (for Enhanced Security)(optional)**

If your objective is to employ SSH key authentication, recognized for its superior security over password authentication, it is necessary to set up an SSH key pair. To facilitate key-based authentication, please execute the ssh-keygen command on your local machine to create the required key pair.

To set up and configure SSH on Ubuntu 22.04, please adhere to the following steps.

**Step 1: Prepare Ubuntu**

Prior to installing SSH on Ubuntu, it is important to make sure that all the appropriate apt packages

are updated to their latest versions. This can be accomplished by executing the given command:

**sudo apt update && sudo apt upgrade**

**Step 2: Install SSH on Ubuntu**

In most cases, the SSH server is included as a standard feature in your Ubuntu system. If the OpenSSH is not included by default, manual installation will be required. To proceed with the installation, type the command indicated here into your terminal.

**sudo apt install openssh-server**

As you install, be sure to select "Yes" for all the system prompts that appear. When the installation is done, go ahead and proceed to the next step to activate the service.

We must first confirm whether our OpenSSH Server is installed on our system. Carry out this command to move forward.

**sudo apt list openssh-server**

## **Step 3: Start SSH**

To initiate the SSH service which is recently installed, make use of the subsequent instructions given below:

If you want to kick off the SSH service which is recently just set up, check out the steps listed below.

**sudo systemctl enable --now ssh**

The inclusion of the **--now** flag will the service and configure it to start automatically when the system powers up.

To determine the operational status of the service, utilize the command indicated below:

To see if the service is up and running, make sure to apply the command listed here:

**sudo systemctl status ssh**

In the output, find the line that reads **Active: active (running);** this signifies that the service is operating without issues.

If you intend to deactivate the service, simply apply the command listed here:

**sudo systemctl disable ssh**

This procedure will terminate the service and block it from activating upon system startup.

**Step 4: Configure the Firewall**

Prior to establishing an SSH connection to the server, it's important to check that the firewall settings are set up correctly. If a firewall is actively running, it is necessary to permit SSH traffic. For those employing UFW (Uncomplicated Firewall), you can enable SSH access by typing in this command.

**sudo ufw allow ssh**

Also, make sure that UFW is configured to start automatically every time the system starts.

**sudo ufw enable**

You can assess its current status by performing the command mentioned below as you are operating with UFW.

**sudo ufw status**

If SSH traffic is blocked, you are able to authorize it by using the command detailed below. then follow the previously mentioned steps for setup.

**sudo ufw allow ssh**

and follow the above-mentioned steps for configuration.

## **Step 5: Connect to the Server**

With the earlier steps behind you, it's time to log into the server utilizing the SSH protocol. You'll need the server's IP address on the other hand the domain name, plus the username of a user that exists on the server. Just type the command into your terminal to get started.

**ssh username@IP\_address**

Alternatively:

**ssh username@domain**

**Important**: To successfully connect to a remote server, SSH must be installed and configured on the remote server and the user's computer from which you make the connection.

To verify whether SSH is functioning, you can connect to your server from a different machine with the running the command:

**ssh your\_username@your\_server\_ip\_address**

Substitute your\_username with your server username and your\_server\_ip\_address with the corresponding IP address of your server.

## **Step 6: Configure SSH**

Now that you've perfectly navigated the first five steps, you're ready to establish a remote connection to the server. However, to enhance the security of this connection, consider altering the default connection port or switching from password authentication to key-based authentication. Making these adjustments will require modifications to the SSH configuration file.

The chief settings for the OpenSSH server are contained within the **sshd\_config** file, which can be stored at **/etc/ssh**.

### Prior to making any changes, it’s important to back up this file.

To safeguard your work, consider duplicating this file prior to making any modifications.

**configuration.sudo cp /etc/ssh/sshd\_config /etc/ssh/sshd\_config.initial**

This precaution will enable you to restore the original version if any mistakes happen after your changes.

### **Edit the Configuration File**

To alter the configuration file, you may conveniently open it with the nano text editor.

**sudo nano /etc/ssh/sshd\_config**

#### **Change the Port**

To modify the configuration file, you can easily open it using the nano text editor. Within the file, adjust the port to a more secure option. It is advisable to select values from the dynamic port range of 49152 to 65535 and to use varied numbers for increased security. For instance, we can set the port value to 48582. This can be accomplished by uncommenting the relevant line in the file and tweak the port number.

#### **Enable Key Authentication**

Moreover, it is recommended to modify the password authentication setting to a more secure key authentication mode. This requires uncommenting the corresponding line and verifying that the value is "**Yes**,"

**PasswordAuthentication no**

To

**PasswordAuthentication Yes**

#### **Disable Root Login**

It is now necessary to implement a prohibition on superuser access to the server by adjusting the corresponding line by setting the following value.

**PermitRootLogin no**

## **Additional Security Settings**

 Below are some recommended configurations that can be implemented to improve the server's security

● **UseDNS**: Verifies whether the hostname corresponds to its associated IP address. The designation "Yes" activates this parameter.

● **PermitEmptyPasswords**: If configured to "No," this option blocks the use of empty passwords during the authentication purposes.

● **MaxAuthTries**: establishes a cap on the number of failed attempts to connect to the server during a single communication session.

● **AllowUsers and AllowGroups** : manage the roster of users and groups that have permission to access the server via SSH.

**# AllowUsers User1, User2, User3**

**# AllowGroups Group1, Group2, Group3**

● **Login GraceTime**: parameter determines the duration allowed for successful user authorization. It is advisable to decrease this value to one-fourth of its current setting.

● **ClientAliveInterval**: setting controls the maximum period of user inactivity. When a user goes beyond this limit, they'll lose their connection.

Once you've made all the changes to the major configuration file, make sure to save your work and exit the editor.

### **Restart the SSH Service**

To put this modification into action, you have to kickoffthe SSH service again.

**sudo systemctl restart ssh**

In the event that the port number has been modified, please connect using the updated port as follows:

**ssh -p port\_number username@IP\_address**

Alternatively, you may use:

**ssh -p port\_number username@domain**

**Step 7: Testing the SSH Connection**

After activating SSH, it is essential to confirm that the service is operational and capable of accepting connections. To conduct this verification, you may assess the connection by utilizing another device. Employ the subsequent command to connect to your Ubuntu system using SSH:

**ssh user\_name@<server\_ip>**

Substitute " user\_name " with your specific username on the Ubuntu system and " server\_ip " with the corresponding hostname or IP address of your Ubuntu system. If SSH settings be correctly established a prompt will appear requesting the user's password, or it will utilize an SSH key if one has been established. After the authentication process is successfully finalized, a shell prompt will be presented, signifying your access to the server.

**Troubleshooting**

If the connection does not succeed, it may indicate that the service is operating. To find out the status, go ahead and execute

**sudo systemctl status ssh.**

If it is found to be inactive, you can restart the service using

**sudo systemctl start ssh**

Firewalls, including ufw, can potentially block SSH traffic. It is vital to confirm that port 22, the designated port for SSH, is available to support incoming connections.

**sudo ufw allow 22**

**sudo ufw enable**

If the server cannot be reached, the underlying issue may be related to the network. Utilize the ping command to ascertain whether the client is able to establish communication with the server.

**ping <server\_ip>**

The file **/etc/ssh/sshd\_config** is responsible for managing SSH configurations. Errors in configuration, including IP limitations or disabled user access, can obstruct connections. Amend the file with care and restart the service to ensure that the alterations are effective.

**Deactivate ssh in Ubuntu**

To deactivate SSH service on an Ubuntu system, proceed with the steps outlined below.

1. The provided command halts the SSH (Secure Shell) service without delay. Following this action, the system will not permit any incoming SSH connections until the service is restarted. The rationale behind this procedure is that the most expedient way to deactivate SSH temporarily is to stop the service.

**sudo systemctl stop ssh**

2. By using this command, you can ensure that the SSH service does not initiate automatically upon system boot. While halting the SSH service can temporarily stop it, there is a possibility that it will automatically restart following a system reboot. Disabling the service guarantees that SSH will remain inactive during subsequent reboots.

**sudo systemctl disable ssh**

3. To ascertain whether the SSH service has been halted, utilize the command that evaluates its present status.

**sudo systemctl status ssh**

In the resulting output, In the output, search for the phrase **"Active: inactive (dead)",** the intention of this message is to confirm that SSH has been deactivated and is no longer capable of accepting connections.

4. If SSH is no longer required on this system, the removal of the package will not only reduce potential vulnerabilities but also free up valuable disk space.

**sudo apt-get remove --purge openssh-server**

This process thoroughly uninstalls the OpenSSH server package and associated configuration files, with the --purge option guarantees the deletion of these files as well.

**Conclusion**

Here’s a comprehensive guide that walks you through the installation and configuration process of SSH on Ubuntu 22.04. It also includes steps to adjust the main configuration file for better security. We anticipate this guide makes it easier for you to establish a secure remote connection to your Ubuntu server.

Similarly, for the deactivation process, Steps 1 and 3 alone are adequate for achieving temporary deactivation and in order to permanently deactivate SSH, execute Steps 1, 2, and 3, ensuring that it is disabled and remains so after the system restarts. In the context of uninstallation, Step 4 is designated for the thorough elimination of SSH when it is deemed unnecessary.