By: Mitchell Anicas

Initial Server Setup with Ubuntu 16.04

Posted April 21, 2016 548.9k GETTING STARTED SECURITY UBUNTU UBUNTU 16.04

Introduction

When you first create a new Ubuntu 16.04 server, there are a few configuration steps that you should take early on as part of the basic setup. This will increase the security and usability of your server and will give you a solid foundation for subsequent actions.

Step One — Root Login

To log into your server, you will need to know your server's public IP address. You will also need the password or, if you installed an SSH key for authentication, the private key for the "root" user's account. If you have not already logged into your server, you may want to follow the first tutorial in this series, How to Connect to Your Droplet with SSH, which covers this process in detail.

If you are not already connected to your server, go ahead and log in as the `root` user using the following command (substitute the highlighted word with your server's public IP address):

```
$ ssh root@your_server_ip
```

Complete the login process by accepting the warning about host authenticity, if it appears, then providing your root authentication (password or private key). If it is your first time logging into the server with a password, you will also be prompted to change the root password.

About Root

The root user is the administrative user in a Linux environment that has very broad privileges. Because of the heightened privileges of the root account, you are actually *discouraged* from using it on a regular basis. This is because part of the power inherent with the root account is the ability to make very destructive changes, even by accident.

The next step is to set up an alternative user account with a reduced scope of influence for day-to-day work. We'll teach you how to gain increased privileges during the times when you need them.

Step Two — Create a New User

Once you are logged in as `root`, we're prepared to add the new user account that we will use to log in from now on.

SCROLL TO TOP

This example creates a new user called "sammy", but you should replace it with a username that you like:

```
# adduser sammy
```

You will be asked a few questions, starting with the account password.

Enter a strong password and, optionally, fill in any of the additional information if you would like. This is not required and you can just hit `ENTER` in any field you wish to skip.

Step Three — Root Privileges

Now, we have a new user account with regular account privileges. However, we may sometimes need to do administrative tasks.

To avoid having to log out of our normal user and log back in as the root account, we can set up what is known as "superuser" or root privileges for our normal account. This will allow our normal user to run commands with administrative privileges by putting the word `sudo` before each command.

To add these privileges to our new user, we need to add the new user to the "sudo" group. By default, on Ubuntu 16.04, users who belong to the "sudo" group are allowed to use the `sudo` command.

As `root`, run this command to add your new user to the *sudo* group (substitute the highlighted word with your new user):

```
# usermod -aG sudo sammy
```

Now your user can run commands with superuser privileges! For more information about how this works, check out this sudoers tutorial.

If you want to increase the security of your server, follow the rest of the steps in this tutorial.

Step Four — Add Public Key Authentication (Recommended)

The next step in securing your server is to set up public key authentication for your new user. Setting this up will increase the security of SCROLL TO TOP requiring a private SSH key to log in.

Generate a Key Pair

If you do not already have an SSH key pair, which consists of a public and private key, you need to generate one. If you already have a key that you want to use, skip to the *Copy the Public Key* step.

To generate a new key pair, enter the following command at the terminal of your **local machine** (ie. your computer):

```
$ ssh-keygen
```

Assuming your local user is called "localuser", you will see output that looks like the following:

```
ssh-keygen output
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/localuser/.ssh/id_rsa):
```

Hit return to accept this file name and path (or enter a new name).

Next, you will be prompted for a passphrase to secure the key with. You may either enter a passphrase or leave the passphrase blank.

Note: If you leave the passphrase blank, you will be able to use the private key for authentication without entering a passphrase. If you enter a passphrase, you will need both the private key *and* the passphrase to log in. Securing your keys with passphrases is more secure, but both methods have their uses and are more secure than basic password authentication.

This generates a private key, `id_rsa`, and a public key, `id_rsa.pub`, in the `.ssh` directory of the *localuser*'s home directory. Remember that the private key should not be shared with anyone who should not have access to your servers!

Copy the Public Key

After generating an SSH key pair, you will want to copy your public key to your new server. We will cover two easy ways to do this.

SCROLL TO TOP

Note: The `ssh-copy-id` method will not work on DigitalOcean if an SSH key was selected during Droplet creation. This is because DigitalOcean disables password authentication if an SSH key is present, and the `ssh-copy-id` relies on password authentication to copy the key.

If you are using DigitalOcean and selected an SSH key during Droplet creation, use option 2 instead.

Option 1: Use ssh-copy-id

If your local machine has the `ssh-copy-id` script installed, you can use it to install your public key to any user that you have login credentials for.

Run the `ssh-copy-id` script by specifying the user and IP address of the server that you want to install the key on, like this:

```
$ ssh-copy-id sammy@your_server_ip
```

After providing your password at the prompt, your public key will be added to the remote user's `.ssh/authorized_keys` file. The corresponding private key can now be used to log into the server.

Option 2: Manually Install the Key

Assuming you generated an SSH key pair using the previous step, use the following command at the terminal of your **local machine** to print your public key (`id_rsa.pub`):

```
$ cat ~/.ssh/id_rsa.pub
```

This should print your public SSH key, which should look something like the following:

```
id_rsa.pub contents
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDBGTO0tsVejssuaYR5R3Y/i73SppJAhme1dH7W2c47d4gOqB4izP0+fRLfvbz/tnXFz4iOP/H6eCV05hqUhF+KYRxt9Y8tVMrpDZR2l75
```

Select the public key, and copy it to your clipboard.

SCROLL TO TOP

To enable the use of SSH key to authenticate as the new remote user, you must add the public key to a special file in the user's home directory.

On the server, as the **root** user, enter the following command to temporarily switch to the new user (substitute your own user name):

```
# su - sammy
```

Now you will be in your new user's home directory.

Create a new directory called `.ssh` and restrict its permissions with the following commands:

```
$ mkdir ~/.ssh
$ chmod 700 ~/.ssh
```

Now open a file in `.ssh` called `authorized_keys` with a text editor. We will use `nano` to edit the file:

```
$ nano ~/.ssh/authorized_keys
```

Now insert your public key (which should be in your clipboard) by pasting it into the editor.

Hit `CTRL-x` to exit the file, then `y` to save the changes that you made, then `ENTER` to confirm the file name.

Now restrict the permissions of the *authorized_keys* file with this command:

```
$ chmod 600 ~/.ssh/authorized_keys
```

Type this command **once** to return to the `root` user:

```
$ exit
```

SCROLL TO TOP

Now your public key is installed, and you can use SSH keys to log in as your user.

To read more about how key authentication works, read this tutorial: How To Configure SSH Key-Based Authentication on a Linux Server.

Next, we'll show you how to increase your server's security by disabling password authentication.

Step Five — Disable Password Authentication (Recommended)

Now that your new user can use SSH keys to log in, you can increase your server's security by disabling password-only authentication. Doing so will restrict SSH access to your server to public key authentication only. That is, the only way to log in to your server (aside from the console) is to possess the private key that pairs with the public key that was installed.

> **Note:** Only disable password authentication if you installed a public key to your user as recommended in the previous section, step four. Otherwise, you will lock yourself out of your server!

To disable password authentication on your server, follow these steps.

As **root** or **your new sudo user**, open the SSH daemon configuration:

```
$ sudo nano /etc/ssh/sshd_config
```

Find the line that specifies `PasswordAuthentication`, uncomment it by deleting the preceding `#`, then change its value to "no". It should look like this after you have made the change:

sshd_config — Disable password authentication

```
PasswordAuthentication no
```

Here are two other settings that are important for key-only authentication and are set by default. If you haven't modified this file before, you *do not* need to change these settings:

SCROLL TO TOP

sshd_config — Important defaults

```
PubkeyAuthentication yes
ChallengeResponseAuthentication no
```

When you are finished making your changes, save and close the file using the method we went over earlier (`CTRL-X`, then `Y`, then `ENTER`).

Type this to reload the SSH daemon:

```
$ sudo systemctl reload sshd
```

Password authentication is now disabled. Your server is now only accessible with SSH key authentication.

Step Six — Test Log In

Now, before you log out of the server, you should test your new configuration. Do not disconnect until you confirm that you can successfully log in via SSH.

In a new terminal on your **local machine**, log in to your server using the new account that we created. To do so, use this command (substitute your username and server IP address):

```
$ ssh sammy@your_server_ip
```

If you added public key authentication to your user, as described in steps four and five, your private key will be used as authentication. Otherwise, you will be prompted for your user's password.

> **Note about key authentication:** If you created your key pair with a passphrase, you will be prompted to enter the passphrase for your key. Otherwise, if your key pair is passphrase-less, you should be logged in to your server without a password.

Once authentication is provided to the server, you will be logged in as your new user.

SCROLL TO TOP

Remember, if you need to run a command with root privileges, type "sudo" before it like this:

```
$ sudo command_to_run
```

Step Seven — Set Up a Basic Firewall

Ubuntu 16.04 servers can use the UFW firewall to make sure only connections to certain services are allowed. We can set up a basic firewall very easily using this application.

Different applications can register their profiles with UFW upon installation. These profiles allow UFW to manage these applications by name. OpenSSH, the service allowing us to connect to our server now, has a profile registered with UFW.

You can see this by typing:

```
$ sudo ufw app list
```

```
Output
Available applications:
  OpenSSH
```

We need to make sure that the firewall allows SSH connections so that we can log back in next time. We can allow these connections by typing:

```
$ sudo ufw allow OpenSSH
```

Afterwards, we can enable the firewall by typing:

```
$ sudo ufw enable
```

SCROLL TO TOP

Type "y" and press ENTER to proceed. You can see that SSH connections are still allowed by typing:

```
$ sudo ufw status
```

```
Output
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
```

If you install and configure additional services, you will need to adjust the firewall settings to allow acceptable traffic in. You can learn some common UFW operations in this guide.

Where To Go From Here?

At this point, you have a solid foundation for your server. You can install any of the software you need on your server now.

By: Mitchell Anicas

Upvote (293) Subscribe Share

SCROLL TO TOP

Related Tutorials

How To Use Filezilla to Transfer and Manage Files Securely on your VPS

Configuration Management 101: Writing Chef Recipes

How To Set Up Nginx Server Blocks (Virtual Hosts) on Ubuntu 16.04

Configuration Management 101: Writing Puppet Manifests

Configuration Management 101: Writing Ansible Playbooks

69 Comments

Leave a comment...

SCROLL TO TOP

Log In to Comment