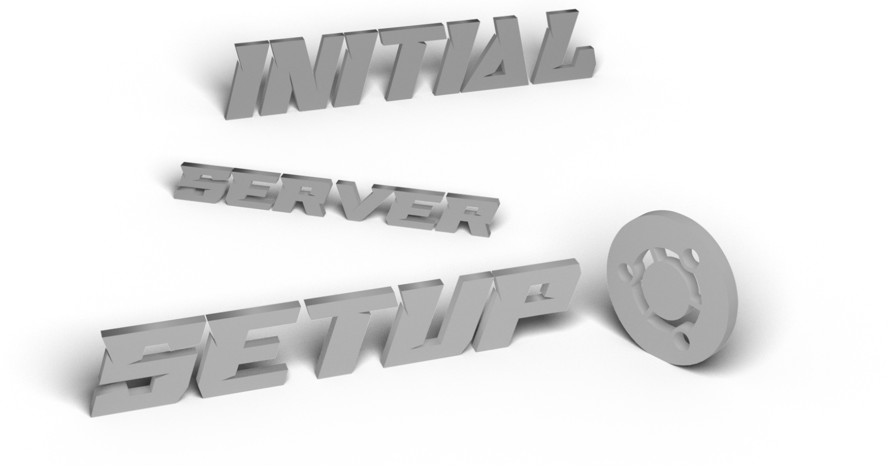

Initial Server Setup with Ubuntu 20.04

When you first create a new Ubuntu 20.04 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 1 - Logging in as Root

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

```
ssh root@your_server_ip
```

Accept the warning about host authenticity if it appears. If you are using password authentication, provide your **root** password to log in. If you are using an SSH key that is passphrase protected, you may be prompted to enter the passphrase the first time you use the key each session. If this is your first time logging into the server with a password, you may 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 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 2 - Creating 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.

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

```
adduser david
```

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 3 - Granting Administrative 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 20.04, 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:

```
usermod -aG sudo david
```

Now, when logged in as your regular user, you can type `sudo` before commands to perform actions with superuser privileges.

Step 4 - Setting Up a Basic Firewall

Ubuntu 20.04 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:

```
ufw app list
```

The output should look like this:

```
Available applications
    OpenSSH
```

Afterwards, we can enable the firewall by typing:

```
ufw enable
```

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

```
ufw status
```

```
Status: active

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

As **the firewall is currently blocking all connections except for SSH**, if you install and configure additional services, you will need to adjust the firewall settings to allow acceptable traffic in.

Step 5 - Enabling External Access for Your Regular User

Now that we have a regular user for daily use, we need to make sure we can SSH into the account directly.

The process for configuring SSH access for your new user depends on whether your server's **root** account uses a password or SSH keys for authentication.

If the Root Account Uses Password Authentication

If you logged in to your **root** account using a password, then password authentication is enabled for SSH. You can SSH to your new user account by opening up a new terminal session and using SSH with your new username:

```
ssh david@your_server_ip
```

After entering your regular user's password, you will be logged in. Remember, if you need to run a command with administrative privileges, type sudo before it like this:

```
sudo command_to_run
```

You will be prompted for your regular user password when using **sudo** for the first time each session (and periodically afterwards).

If the Root Account Uses SSH Key Authentication

If you logged in to your **root** account using SSH keys, then password authentication is disabled for SSH. You will need to add a copy of your local public key to the new user's **~/.ssh/authorized_keys** file to log in successfully.

Since your public key is already in the **root** account's **~/.ssh/authorized_keys** file on the server, we can copy that file and directory structure to our new user account in our existing session.

The simplest way to copy the files with the correct ownership and permissions is with the **rsync** command. This will copy the **root** user's **.ssh** directory, preserve the permissions, and modify the file owners, all in a single command. Make sure to change the highlighted portions of the command below to match your regular user's name:

```
rsync --archive --chown=david:david ~/.ssh /home/david
```

Now, open up a new terminal session and using SSH with your new username:

```
ssh david@your_server_ip
```

You should be logged in to the new user account without using a password. Remember, if you need to run a command with administrative privileges, type **sudo** before it like this:

```
sudo command_to_run
```

You will be prompted for your regular user password when using **sudo** for the first time each session (and periodically afterwards).