Initial Server Setup with Ubuntu 14.04

Posted April 17, 2014 2m GETTING STARTED LINUX BASICS INITIAL SERVER SETUP UBUNTU

By: Justin Ellingwood

Not using **Ubuntu 14.04**? Choose a different version:

Introduction

When you first create a new Ubuntu 14.04 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 and the password 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):

```
local$ ssh root@SERVER_IP_ADDRESS
```

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.

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

```
# adduser demo
```

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 "super user" 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 14.04, 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):

```
# gpasswd -a demo sudo
```

Now your user can run commands with super user privileges! For more information about how this works, check out this sudoers 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 your server by 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):

```
local$ 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.

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:

```
local$ ssh-copy-id demo@SERVER_IP_ADDRESS
```

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`):

```
local$ 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+fRL
```

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

Add Public Key to New Remote User

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 switch to the new user (substitute your own user name):

```
# su - demo
```

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
```

Now you may SSH login as your new user, using the private key as authentication.

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

Step Five — Configure SSH Daemon

Now that we have our new account, we can secure our server a little bit by modifying its SSH daemon configuration (the program that allows us to log in remotely) to disallow remote SSH access to the **root** account.

Begin by opening the configuration file with your text editor as root:

```
# nano /etc/ssh/sshd_config
```

Next, we need to find the line that looks like this:

/etc/ssh/sshd_config (before)

```
PermitRootLogin yes
```

Here, we have the option to disable root login through SSH. This is generally a more secure setting since we can now access our server through our normal user account and escalate privileges when necessary.

Modify this line to "no" like this to disable root login:

/etc/ssh/sshd_config (after)

```
PermitRootLogin no
```

Disabling remote root login is highly recommended on every server!

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`).

Step Six -- Reload SSH

Now that we have made our change, we need to restart the SSH service so that it will use our new configuration.

Type this to restart SSH:

```
# service ssh restart
```

Now, before we log out of the server, we should **test** our new configuration. We do not want to disconnect until we can confirm that new connections can be established successfully.

Open a **new** terminal window on your local machine. In the new window, we need to begin a new connection to our server. This time, instead of using the root account, we want to use the new account that we created.

For the server that we showed you how to configure above, you would connect using this command. Substitute your own user name and server IP address where appropriate:

```
local$ ssh demo@SERVER_IP_ADDRESS
```

Note: If you are using PuTTY to connect to your servers, be sure to update the session's *port* number to match your server's current configuration.

You will be prompted for the new user's password that you configured. After that, you will be logged in as your new user.

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

```
$ sudo command_to_run
```

If all is well, you can exit your sessions by typing:

```
$ exit
```

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.

If you are not sure what you want to do with your server, check out the next tutorial in this series for Additional Recommended Steps for New Ubuntu 14.04 Servers. It covers things like basic firewall settings, NTP, and swap files. It also provides links to tutorials that show you how to set up common web applications. You may also want to check out this guide to learn how to enable `fail2ban` to reduce the effectiveness of brute force attacks.

If you just want to explore, take a look at the rest of our community to find more tutorials. Some popular ideas are configuring a LAMP stack or a LEMP stack, which will allow you to host websites.

By: Justin Ellingwood

Tutorial Series

Introduction to Nginx and LEMP on Ubuntu 14.04

This tutorial series helps sysadmins set up a new web server using the LEMP stack, focusing on Nginx setup with virtual blocks. This will let you serve multiple websites from one Droplet. You'll start by setting up your Ubuntu 14.04 server and end with multiple virtual blocks set up for your websites. An Nginx configuration guide is included at the end for reference.

Show Tutorials

New Ubuntu 14.04 Server Checklist

When creating a new Ubuntu 14.04 server, there are some basic steps that you should take to ensure that your server is secure and configured properly. This tutorial series covers connecting to your server and general security best practices, and provides links to articles that will help you start running your own web server or application.

Show Tutorials

Related Tutorials

How to Get Started with FreeBSD

How To Set Up vsftpd for a User's Directory on Debian 9

How To Set Up Time Synchronization on Debian 9

Initial Server Setup with Debian 9

How to Set Up SSH Keys on Debian 9

175 Comments

Leave a comment...

rimouski *April 20, 2014*

1 When I try to restart the SSH server after some modifications I get this error:

WARNING: Usage of "server" is deprecated, it has been renamed to "rspserver"!
Starting service Echo...
Echo Server - Version 1.0
==========

General Parameters:
Pool Handle = EchoPool
Reregistration Interval = 30.000s
Local Addresses = { all }
Runtime Limit = off
Policy Settings
Policy Type = RoundRobin
Load Degradation = 0.000%
Load DPF = 0.000%
Weight = 0
Weight DPF = 0.000%
20-Apr-2014 01:51:53.0852: P3130.7f57bfb19780@stuff rserpoolsocket.c:354 doRegistration()
20-Apr-2014 01:51:53.0857: Error: (Re-)Registration failed: no registrar available
Registration:
Identifier = $62292d97

Plus, my shell prompt become's locked and I have to reconnect.

This is with a fresh copy of Ubuntu 14.04 installed minutes ago.

Fred

---

rick599215 *April 20, 2014*

1 It seems that denyhosts is marked unmaintained for 14.04.

---

dustinmatlock *April 21, 2014*

0 I think you meant to say under Step Four:

"replacing "demo" with the user you created:"

---

asb MOD *April 21, 2014*

0 @Dustin Fixed. Thanks for catching that!

asb MOD *April 21, 2014*

0 @Rick Roberts: Right, denyhosts has been dead upstream for awhile, and Ubuntu and Debian have decided to stop providing it. Check out:

http://askubuntu.com/questions/433924/package-denyhosts-in-ubuntu-trusty-tahr-is-deleted-temporary-or-forever

You should use fail2ban. I'll remove the link to denyhosts above. Thanks!

---

tamara *April 23, 2014*

4 If anyone following step 5 gets stuck with a "Connection refused" error (like I did), there may be a typo in your config file. While still logged in as root in your other terminal window, you can run the following to test the ssh config file for issues:

sshd -T

If there is an issue with a config file, it will tell you which one and on what line.

If you logged out for some reason, you can use DO's console access to log in and fix any issues.

Cheers

---

lcividin *October 30, 2014*

1 Thanks for posting this!! (sshd -T)
I had a typo myself Allowuser should of been Allowusers and I couldn't see it. This was driving me mad as everything looked correct the ssh service would start but I couldn't get anything from netstat or ps. I kept looking at my configs just couldn't see the typo lol!
I read your post and went oh, ran the command and bam there it was! thwarted by a single character haha!

Thanks again great very happy :)

Ant1 *July 15, 2015*

0 Thank you! Luckily I came across your post as soon as I encountered the issue.

---

webmaster645665 *May 1, 2014*

0 I have followed this tutorial and it is all working, but now I have followed the sftp tutorial too and it keeps checking port 22. With this tutorial I changed that port "to something in between 1025 and 65536", so my question is: how do I know get SFTP to work?

jellingwood MOD *May 2, 2014*

0 Piet:

If you've changed the port that SSH operates on, you can tell the `sftp` command or your file transfer client the new port you selected.

If you are running from the command line, you can specify a non-default port like this. If the port you selected is 4444, the command would look like:

`sftp -P 4444 username@server.com`

If you are using an (S)FTP client to connect to the server, you can input the new port in the options. For instance, in Filezilla, there is a field on the right-hand side labeled "Port" as you can see here:

https://i.imgur.com/4FBl4vX.png

Let us know if that works for you or not.

---

drphilbobaggins *May 5, 2014*

0 I may be missing something, but if the point is to make it more secure by removing roots access, and giving another user the permissions... haven't we just chased our own tail?

We're exactly where we started, except that it's now also a username they need to know.

---

asb MOD *May 5, 2014*

1 @drphilbobaggins: Having to guess the username is surprisingly helpful when it comes to server security. Run a server for long enough, and you'll see many brute force attempts to SSH into the root account. So if you don't want to disable the root user, at least create a new user and set PermitRootLogin to no in your SSH configuration.

---

clarke.hamilton *May 11, 2014*

0 I am using putty to log into my server, I followed the tutorial, added user changed port and log in with putty under new settings I now receive the message:
"server refused our key"

I am then prompted for pass, for the new user I enter it and gain access. I am not sure what I have missed in the tutorial that is stopping the SSH key from authenticating.

This authenticated when it was on default 22 and login as root.

---

iamjsonkim *May 11, 2014*

1 Went through this guide perfectly without any problem. Thank you for the write up Justin.

clarke.hamilton *May 11, 2014*

0 It works, searched for putty tutorials, I had to add my public key for my new user /.ssh/authorized_keys and chmod 700 and 600

adsf *May 12, 2014*

0 Please add in the tutorial
username ALL=(ALL) NOPASSWD: ALL
for those that dont want to type password for sudo each time.
It is not practical to type password each time
thanks

kamaln7 MOD *May 13, 2014*

0 @adsf: I wouldn't recommend doing that. It's insecure to leave sudo access unprotected, it's actually more secure to type out your password every one in a while.

tom593555 *May 14, 2014*

0 @adsf: If typing the password is an annoyance for you, type "sudo su" to temporarily log in as root. This is a much better option than leaving sudo access unprotected.

timothybernerslee *May 26, 2014*

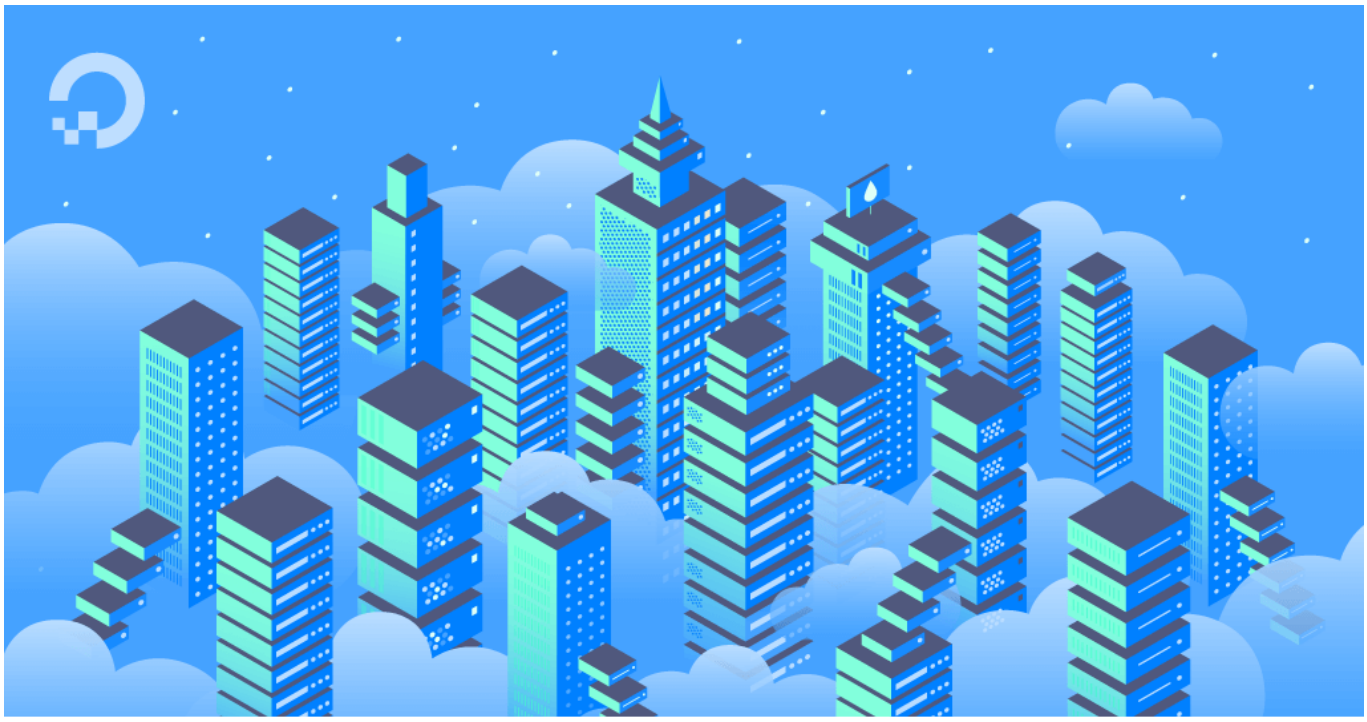
0 When I use FileZilla and try delete index.php from /var/www/html/ I get - permission denied. How send sudo to ftp-client?

kamaln7 MOD *May 27, 2014*

0 @timothybernerslee: You can't use sudo or connect as root using FTP. Use SFTP instead -- it's much more secure (while the name is very similar to "FTP", it's not related to the FTP protocol at all): https://www.digitalocean.com/community/articles/how-to-use-filezilla-to-transfer-and-manage-files-securely-on-your-vps.

Make sure your user has write permissions to index.php so you can delete it:

```
sudo chown youruser /var/www/html/index.php
```



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

by Pablo Carranza

This article will teach you how to use Filezilla to transfer and manage files securely on your VPS.

---

web *June 12, 2014*

0 After going through these steps I'm finding that ssh access is requiring a password to login (using the new user btw and not root as that is of course disabled) , in this case ssh keys were setup during droplet creation, is there another step to take to rectify this so no password is required for ssh with this new user?

Note: initially root@droplet-ip ssh is working out of the box with no password required on this same laptop/droplet combo.

Thanks!

LiewCF *December 26, 2016*

1 I had the same problem and it has been solved.

Apparently, the "authorized*keys" file's owner need to be the account username, NOT root. Login as root to change the file's owner "chown user:user authorized*keys" will solve the problem.

---

asb MOD *June 12, 2014*

2 You need to make sure that your public key is in the authorized_keys file for your user. Run:

```
cat ~/.ssh/id_rsa.pub | ssh user@your.ip.address "cat >> ~/.ssh/authorized_keys"
```

bijuoos *December 11, 2014*

0 I got the Public Key in section 4. How can I obtain my private key? That not mentioned here and its a most confusing things newbies like me.

asb MOD *December 11, 2014*

0 @bijuoosmail Running the `ssh-keygen` command will produce two files. Your private key, `id_rsa`, which should stay on your local computor, and a public key that is copied to the server `id_rsa.pub`

Load More Comments