Setup Webserver using a VPS with Ubuntu 18 LTS

Presented By Harley @ HF DESIGNS

# Step 1 – Create a Digital Ocean Account

We need to setup an account with Digital Ocean so we can get started with our webserver. Use my referral code you can get $10 in credit. That is enough hosting for two months.

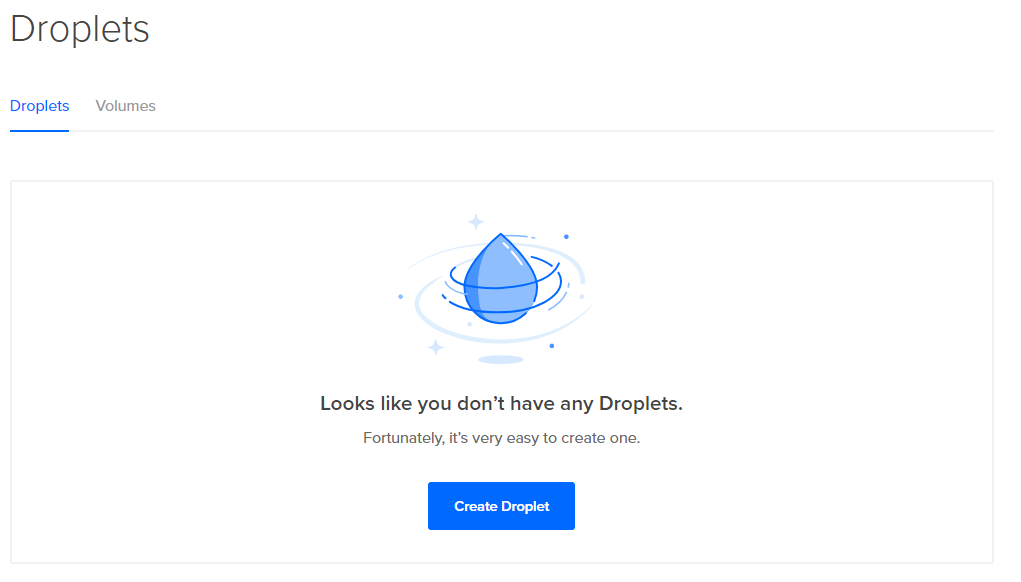
Referral Link: <https://m.do.co/c/1c3a946198ad>

Go ahead and create an account and get to the home page.

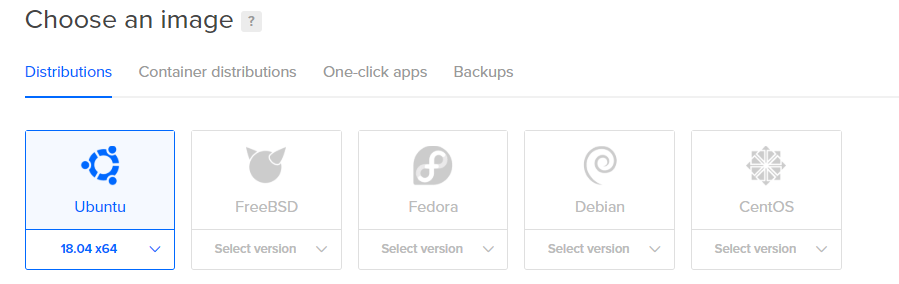
# Step 2 – Create a Droplet



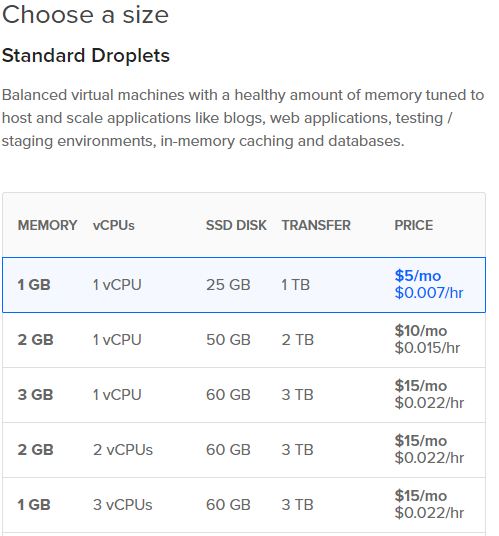
Top Right corner of the webpage you will see a green **CREATE** button. Click/Hover over the button and then select Droplet from the menu. This will bring you to the droplet creation page.



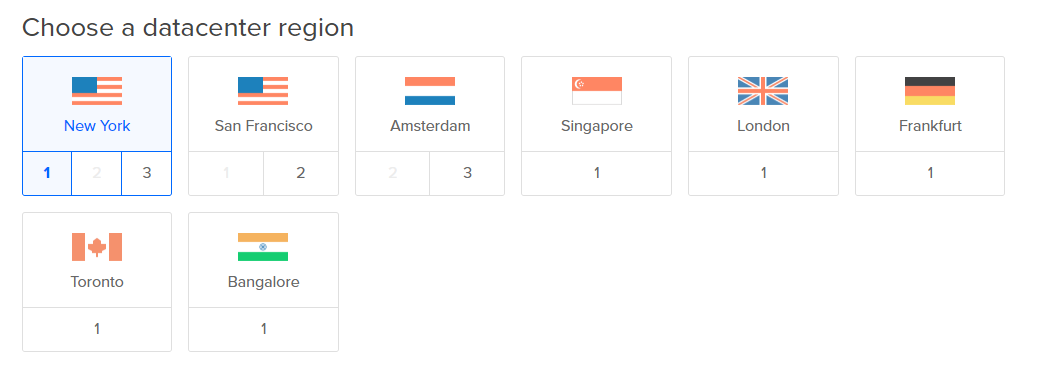
Or you can click **Create Droplet** in the middle of the page.



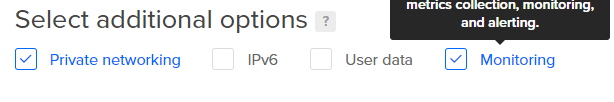
Select Ubuntu 18.04 x64 as the image.



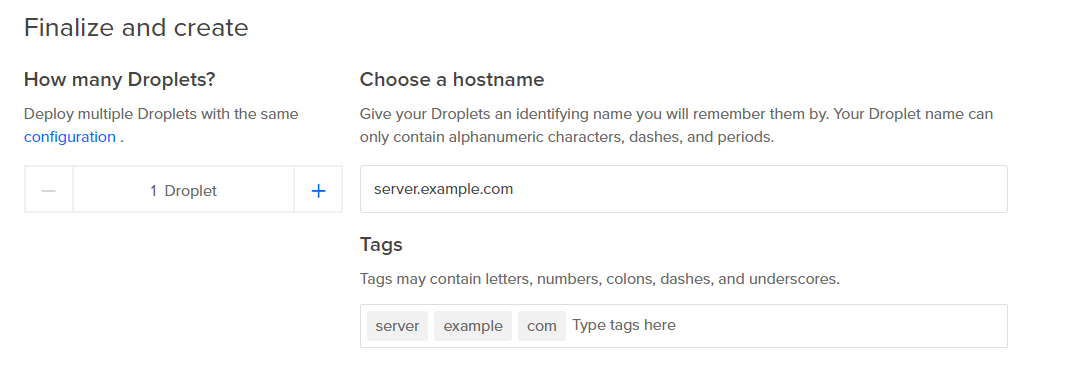
Select the top option from the Standard Droplets.



Choose the region you are located to host the droplet.

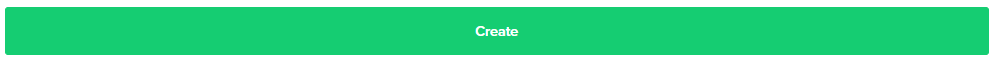


Make sure these are selected.



As a hostname it’s usually best to do like **hbfserver.harleybfrank.com**. It’s understandable if you don’t have a domain name yet so you can do something like **harleytest.example.com** or **harleyserver.example.com**. As long as it looks valid. Do not exceed 255 characters.

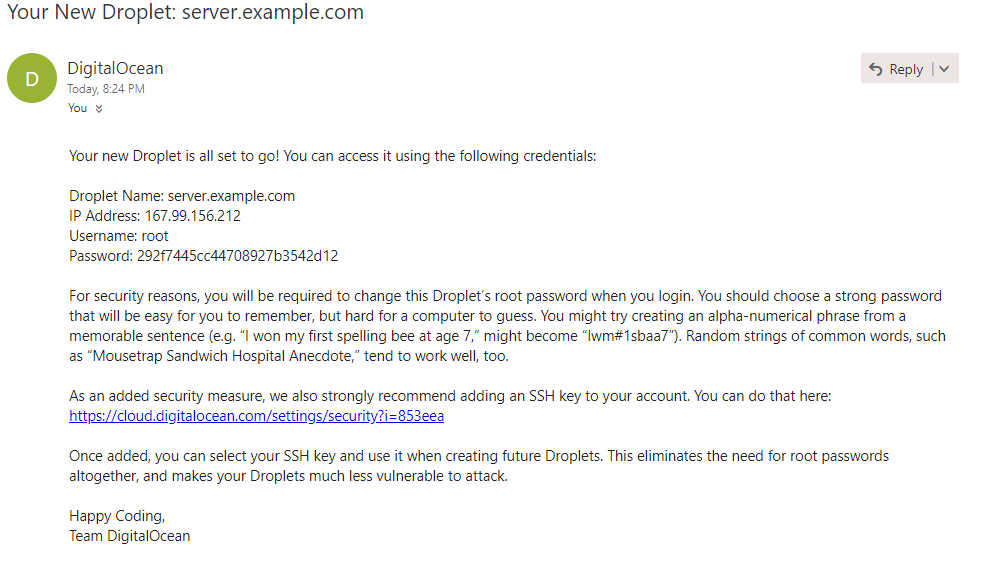
**NOTE:** this is not a valid web address so you will not be able to browse to it unless you use the IP Address given to you. You can always add a domain name later.



It’s time to create.



After you click create you will be brought to the dashboard where it shows you the progress of the creation. Once it is created you will receive and email with your username and password to access via remote login.



# Step 3 – Install Some Software

**Putty SSH Client and Putty Gen**

Programs that allow us to communicate with our server and create a secure login.

Client: <https://the.earth.li/~sgtatham/putty/latest/w64/putty-64bit-0.70-installer.msi>

Gen: <https://the.earth.li/~sgtatham/putty/latest/w64/puttygen.exe>

**FileZilla**

Program that allows us to upload files directly to our server.

<https://filezilla-project.org/>

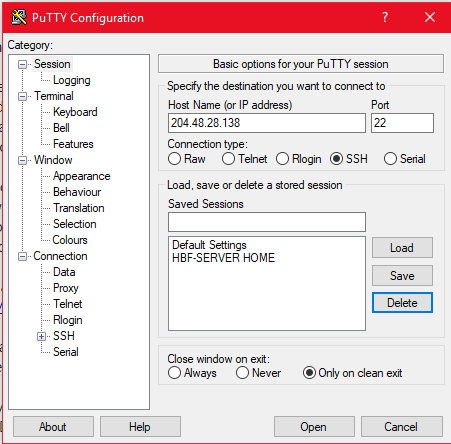
**Notepad++**

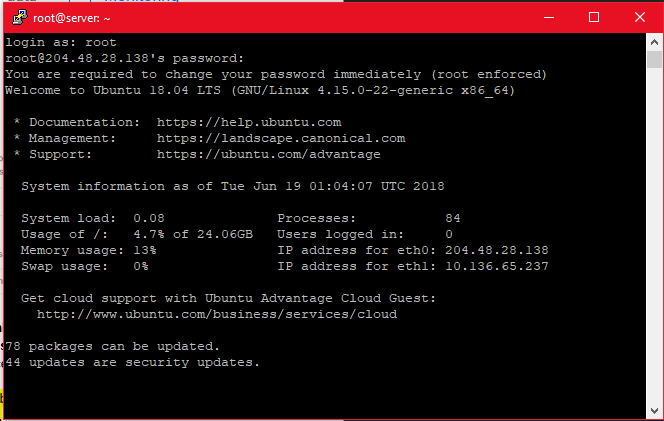
A program that is like notepad but interprets code very well.

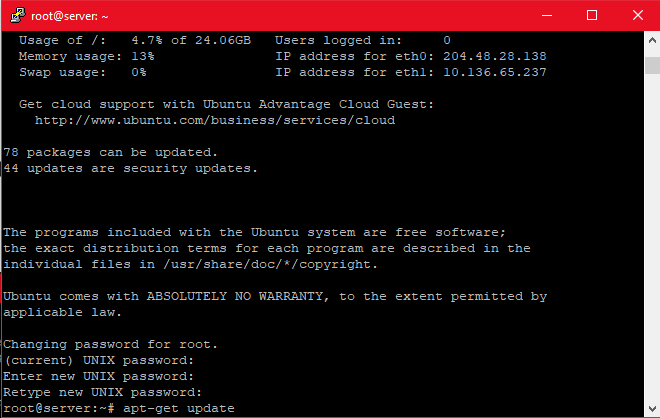
<https://notepad-plus-plus.org/download/v7.5.6.html>

# Step 4 – Connect to the Droplet using Putty

Open Putty and type in your server’s IP Address







Once you have logged in and changed your password it is time to continue to the next step.

# Step 5 – Install LAMP STACK

**L - Linux**

**A – Apache2**

**M - MySQL**

**P – PHP**

This will instantly turn our server into a webserver by installing those components listed above.

|  |
| --- |
| **sudo apt-get update**  **sudo apt-get install lamp-server^** |

**NOTE:** The “sudo apt-get update” will refresh the repositories that are stored on the server and update them so we make sure we are pulling the latest when installing software.

This will take some time, but after it completes you should be able to go to <http://ipaddress> and get a ‘IT WORKS’ page.

# Step 6 – Setup Directories and Attach our Domains

If you do not have a domain then you can just use the default directory already created. You can skip ahead to step 7. If you do have a domain, then you will need to contact your domain registrar to have it pointed to your droplet. Once it has been pointed you can view it like any other website.

First, we need to create a new user for us to access as we don’t want to use the root account. The reason is that the root is a back door to the system and usually blocked off in most installations. However, Digital Ocean uses that for us in case we get locked out. Later on in this tutorial, I will be showing you how to restrict access and using secured keys.

**sudo adduser [new username]**

Enter in a new password twice and fill out the information as necessary.

Now we need to create directories for our web addresses and get the permissions set.

If we use the **cat /etc/group** command, we can see that apache2 created a group for use to add our users in to modify data. So, our documents are still owned by root. We need to make sure that we have access.

Use the following command to add our user account to that group.

**sudo adduser [your username] www-data**

We also need to give that account some root permissions.

**sudo adduser [username] sudo**

**sudo adduser [username] root**

Now we need to create two folders. One is the servername.yourdomain.com and the other is yourdomain.com. Please replace accordingly.

**sudo mkdir -p /var/www/html/servername.yourdomain.com/**

**sudo mkdir -p /var/www/html/yourdomain.com/**

Now to set the permissions correctly:

**sudo chown root:www-data -R /var/www/html/servername.yourdomain.com/**

**sudo chown root:www-data -R /var/www/html/yourdomain.com/**

**sudo chmod -R 775 /var/www/html/servername.yourdomain.com/**

**sudo chmod -R 775 /var/www/html/yourdomain.com/**

Inside these folders we need to create index.html files to test if we configured everything correctly.

We need to create two files and save them in the proper directory. Remember that **Control+O** is to save and **Control+X** is to exit the text editor.

**sudo nano /var/www/html/servername.yourdomain.com/index.html**

**sudo nano /var/www/html/yourdomain.com/index.html**

Replace **[domain name]** with the proper address.

<html>

<head>

<title>Welcome to [domain name]!</title>

</head>

<body>

<h1>Success! The [domain name] virtual host is working!</h1>

</body>

</html>

Save and exit.

Now to copy the default config file to a new config file for our domain setup. We will need to do this for every domain we set up with directories. Replace the names accordingly.

**sudo cp /etc/apache2/sites-available/000-default.conf /etc/apache2/sites-available/[replace].conf**

**sudo nano /etc/apache2/sites-available/[domain].conf**

Should look something like this:

<VirtualHost \*:80>

ServerAdmin webmaster@localhost

DocumentRoot /var/www/html

ErrorLog ${APACHE\_LOG\_DIR}/error.log

CustomLog ${APACHE\_LOG\_DIR}/access.log combined

</VirtualHost>

Make sure to replace the ServerAdmin with your real email address. Also replace DocumentRoot with the correct site directory.

Add ServerName [domain name] below ServerAdmin. Also add ServerAlias [www.[domain](http://www.[domain)] under DocumentRoot so that the site is also browseable when someone types in www. in front of the domain.

Scroll down, and just before </VirtualHost> add this:

<Directory /var/www/html/[domain]/>

Options -Indexes +FollowSymLinks +MultiViews

AllowOverride All

Require all granted

</Directory>

Let's enable the config files now. Make sure to replace the [domain] with the proper name.

**sudo a2ensite [domain].conf**

The default config file is still enabled. For this to properly work, we need to disable that config file.

**sudo a2dissite 000-deafult.conf**

Now we need to use **sudo service apache2 reload** to activate the new configuration. Then use **sudo service apache2 restart** to restart all apache2 services.

# Step 7 – Secure MySQL Installation

This is intended to set our Database system into production mode.

**mysql\_secure\_installation**

You will be asked to enter the password you set for the MySQL root account. Next, you will be asked if you want to configure the VALIDATE PASSWORD PLUGIN.

**Warning:** Enabling this feature is a judgment call. If you enable this, passwords which don't match the specified criteria will be rejected by MySQL with an error. This will cause issues if you use a weak password in conjunction with software which automatically configures MySQL user credentials, such as the Ubuntu packages for phpMyAdmin. It is safe to leave validation disabled, but you should always use strong, unique passwords for database credentials.

*VALIDATE PASSWORD PLUGIN can be used to test passwords*

*and improve security. It checks the strength of password*

*and allows the users to set only those passwords which are*

*secure enough. Would you like to setup VALIDATE PASSWORD plugin?*

*Press Y for Yes, any other key for No:*

**Go ahead and answer no.**

Because the LAMP-STACK already installed MySQL, we haven’t set up a password. Let’s go ahead and set up a root level password for our database system. Even though this is local use only, we need a secure password. Anything between 16 and 32 characters with numbers, letters, and symbols should be fine. When you need the database access you can use Phpmyadmin and the root account to create new accounts for access.

For the rest of the questions, you should press Y and hit the Enter key at each prompt. This will remove some anonymous users and the test database, disable remote root logins, and load these new rules so that MySQL immediately respects the changes we have made.

We need to setup an account for us to use when we are in PHPmyadmin. Replace the USERNAME and PASSWORD fields accordingly.

**% mysql --user=root mysql**

**CREATE USER 'username'@'localhost' IDENTIFIED BY 'password';**

**GRANT ALL PRIVILEGES ON \*.\* TO 'username'@'localhost' WITH GRANT OPTION;**

**FLUSH PRIVILEGES;**

You can use **Control+Z** to exit out of the MySQL interface.

At this point, your database system is now set up and we can move on.

# Step 8 – Install Phpmyadmin and Secure Instance

phpMyAdmin is a web program that allows us to modify our databases in MySQL. Since we have apache we can easily install it with console commands. Ubuntu's default repositories already contain the needed files so we can just use the following commands:

**sudo apt-get update**

**sudo apt-get install phpmyadmin php-mbstring php-gettext**

This will ask you a few questions to configure your installation correctly.

**Warning:** When the first prompt appears, apache2 is highlighted, but not selected. If you do not hit Space to select Apache, the installer will not move the necessary files during installation. Hit Space, Tab, and then Enter to select Apache.

* For the server selection, choose apache2.
* Select yes when asked whether to use dbconfig-common to set up the database
* You will be prompted for your database administrator's password
* You will then be asked to choose and confirm a password for the phpMyAdmin application itself

The installation process adds the phpMyAdmin Apache configuration file into the **/etc/apache2/conf-enabled/** directory, where it is automatically read.

The only thing we need to do is explicitly enable the PHP mcrypt and mbstring extensions, which we can do by typing:

**sudo phpenmod mcrypt**

**sudo apt-get install mcrypt php7.0-mcrypt**

**sudo phpenmod mbstring**

Afterwards, you'll need to restart Apache for your changes to be recognized:

**sudo service apache2 restart**

You can now access the web interface by visiting your server's domain name or public IP address followed by /phpmyadmin:

<http://domain.com/phpmyadmin> or <http://ip/phpmyadmin>

We were able to get our phpMyAdmin interface up and running easily. However, we are not done yet. Because of its ubiquity, phpMyAdmin is a popular target for attackers. We need to take extra steps to prevent unauthorized access.

One of the easiest way of doing this is to place a gateway in front of the entire application. We can do this using Apache's built-in .htaccess authentication and authorization functionalities.

First, we need to enable the use of .htaccess file overrides by editing our Apache configuration file.

We will edit the linked file that has been placed in our Apache configuration directory:

**sudo nano /etc/apache2/conf-available/phpmyadmin.conf**

We need to add an AllowOverride All directive within the <Directory /usr/share/phpmyadmin> section of the configuration file, like this:

*<Directory /usr/share/phpmyadmin>*

*Options FollowSymLinks*

*DirectoryIndex index.php*

*AllowOverride All*

*. . .*

When you have added this line, save and close the file. Use Control+O to save and Control+X to exit.

To implement the changes you made, restart Apache:

**sudo service apache2 restart**

Now that we have enabled .htaccess use for our application, we need to create one to implement some security.

For this to be successful, the file must be created within the application directory. We can create the necessary file and open it in our text editor with root privileges by typing:

**sudo nano /usr/share/phpmyadmin/.htaccess**

Within this file, we need to enter the following information:

*AuthType Basic*

*AuthName "Restricted Files"*

*AuthUserFile /etc/phpmyadmin/.htpasswd*

*Require valid-user*

Let's go over what each of these lines mean:

* AuthType Basic: This line specifies the authentication type that we are implementing. This type will implement password authentication using a password file.
* AuthName: This sets the message for the authentication dialog box. You should keep this generic so that unauthorized users won't gain any information about what is being protected.
* AuthUserFile: This sets the location of the password file that will be used for authentication. This should be outside of the directories that are being served. We will create this file shortly.
* Require valid-user: This specifies that only authenticated users should be given access to this resource. This is what actually stops unauthorized users from entering.

When you are finished, save and close the file. Use Control+O to save and Control+X to exit.

Now that we have specified a location for our password file through the use of the AuthUserFile directive within our .htaccess file, we need to create this file.

We need an additional package to complete this process. We can install it from our default repositories:

**sudo apt-get install apache2-utils**

Afterward, we will have the htpasswd utility available.

The location that we selected for the password file was "/etc/phpmyadmin/.htpasswd". Let's create this file and pass it an initial user by typing:

**sudo htpasswd -c /etc/phpmyadmin/.htpasswd [username]**

Replace [username] with the user you are creating it for.

You will be prompted to select and confirm a password for the user you are creating. Afterwards, the file is created with the hashed password that you entered.

If you want to enter an additional user, you need to do so without the -c flag, like this:

**sudo htpasswd /etc/phpmyadmin/.htpasswd [additionaluser]**

Now, when you access your phpMyAdmin subdirectory, you will be prompted for the additional account name and password that you just configured.

# Step 9 – Setup a Mail Transfer Agent

Now we need a way for our server to send us emails when we need it to. Follow this link to setup using Gmail if you want to. If you have your own mail host like GoDaddy or Microsoft, then you will need to get the settings from your provider. I am going to use my own hosting service for this. If you chose to use your Gmail account, make sure you have less secure apps turned on. If you have two factor authentication on, you may need an App Password. Gmail also only allows 500 emails per day to be sent this method.

<https://help.ubuntu.com/community/EmailAlerts>

**sudo apt-get install ssmtp**

**sudo nano /etc/ssmtp/ssmtp.conf**

Make the necessary changes, then save and close. Use Control+O to write out and Control+X to exit.

**If you are using your own email domain service:**

root=[your email address]

mailhub=[the mail host address]

AuthUser=[authorized email account]

AuthPass=[authorized email password]

UseTLS=YES

UseSSL=YES

rewriteDomain=[the url of your email address]

**If you are using Gmail:**

<https://wiki.archlinux.org/index.php/SSMTP>

Install mail client. **sudo apt-get install mailutils**

echo "We are, we are. The Youth of the Nation!" | mail -s "Test Subject" [your email]

# Step 10 – Setup SFTP

We need to set up a way to access our server remotely to upload and download files. This is an extremely useful tool to have on any webserver.

Let's start by installing the client:

**sudo apt-get install vsftpd**

Now we need to configure it:

**sudo nano /etc/vsftpd.conf**

Make sure these are not commented out [#] and they are displayed like below. Go ahead find these and make the changes.

* anonymous\_enable=NO
* local\_enable=YES
* write\_enable=YES

# Step 11 – Setup DenyHosts

DenyHosts is a python program that automatically blocks SSH attacks by adding entries to /etc/hosts.deny. DenyHosts will also inform Linux administrators about offending hosts, attacked users and suspicious logins.

Open a Terminal and enter the following:

s**udo apt-get install denyhosts**

After installation edit the configuration file /etc/denyhosts.conf and change the email, and other settings as required.

To edit the admin email settings, open a terminal window and enter:

**sudo nano /etc/denyhosts.conf**

Change the following values as required on your server:

ADMIN\_EMAIL = root@localhost

SMTP\_HOST = localhost

SMTP\_PORT = 25

#SMTP\_USERNAME=foo

#SMTP\_PASSWORD=bar

SMTP\_FROM = DenyHosts nobody@localhost

#SYSLOG\_REPORT=YES

# Step 12 – Setup Fail2Ban

Fail2ban is more advanced than DenyHosts as it extends the log monitoring to other services including SSH, Apache, Courier, FTP, and more. Fail2ban scans log files and bans IPs that show the malicious signs -- too many password failures, seeking for exploits, etc. Generally, Fail2Ban then used to update firewall rules to reject the IP addresses for a specified amount of time, although any arbitrary other action could also be configured.

Out of the box Fail2Ban comes with filters for various services (apache, courier, ftp, ssh, etc).

Open a Terminal and enter the following:

**sudo apt-get install fail2ban**

After installation edit the configuration file /etc/fail2ban/jail.local and create the filter rules as required.

To edit the settings open a terminal window and enter:

**sudo nano /etc/fail2ban/jail.conf**

Activate all the services you would like fail2ban to monitor by changing enabled = false to enabled = true

For example, if you would like to enable the SSH monitoring and banning jail, find the line below and change enabled from false to true. That's it.

[sshd]

enabled = true

port = ssh

filter = sshd

logpath = /var/log/auth.log

maxretry = 3

If you would like to receive emails from Fail2Ban if hosts are banned change the following line to your email address.

destemail = root@localhost

Look under ACTIONS.

Good instructions on how to configure fail2ban and create the various filters can be found on [HowtoForge](http://www.howtoforge.com/perfect-server-ubuntu-11.10-ispconfig-3-p5) When done with the configuration of Fail2Ban restart the service with :

**sudo service fail2ban restart**

# Step 13 – Firewall Setup

UFW is already installed on the Ubuntu System by Default. It's a configurator for the iptables system.

**sudo ufw app list** - shows available applications

**sudo ufw app info "[app name]"** - shows application info

**Config Section**

**sudo ufw default deny**

**sudo ufw allow in "Apache Full"**

**sudo ufw allow OpenSSH**

**sudo ufw allow ssh**

**sudo ufw allow ftp**

**sudo ufw allow smtp**

**sudo ufw enable**

**sudo ufw status** - to show what's enabled

# Step 14 – Update AppArmor

Apparmor is included by default, but we can install some extra profiles to help increase our security on our server.

**sudo apt-get install apparmor apparmor-profiles**

**sudo apparmor\_status**

# Step 15 - Auto Update Permissions for Folders on the Server

I have come to know that most of the files uploaded to the server change frequently with permissions. This script will update the permissions every 5 minutes so that way there is no view issues. Now if you are running WordPress or something like that, you may need to modify this to apply separate permissions on those certain files.

**sudo nano** **/.scripts/webupdate.sh**

**#!/bin/bash**

**chown -R root:www-data /var/www/html**

**chmod -R 775 /var/www/html**

Use Control+O to save and Control+X to exit.

**sudo crontab -e**

**\*/5 \* \* \* \* /.scripts/webupdate.sh**

Use Control+O to save and Control+X to exit.

Now you may need to update permissions on the reports and scripts files as you go along further so it’s best if you create a script now and then run it when you run into complications later using the bash command.

**sudo nano /.scripts/general\_update.sh**

**#!/bin/bash**

**sudo chown root:sudo -R /.scripts/**

**sudo chmod -R 775 /.scripts/**

**sudo chown root:sudo -R /.reports/**

**sudo chmod -R 775 /.reports/**

Use Control+O to save and Control+X to exit.

When you want to run it just use this:

**sudo bash /.scripts/general\_update.sh**

# Step 16 – Secure SSH

We will create an account strictly for SSH access and SFTP access. You can either add a key or just change the port and make a super secure password.

The best way to secure SSH is to use public/private key based login. See SSH/OpenSSH/Keys

If you must use password authentication, the easiest way to secure SSH is to disable root login and change the SSH port to something different than the standard port 22. Before disabling the root login create a new SSH user and make sure the user belongs to the admin group (see step 4. below regarding the admin group). If you change the SSH port keep the port number below 1024 as these are privileged ports that can only be opened by root or processes running as root. If you change the SSH port also open the new port, you have chosen on the firewall and close port 22.

Open a Terminal Window and enter:

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

Change or add the following and save.

**Port <ENTER YOUR PORT>**

**Protocol 2**

**PermitRootLogin no**

Next need to make sure it is in our firewall now:

**sudo ufw allow in [new port]**

**sudo service ssh restart**

If you want to secure your login further using secured keys, follow the steps below.

Let's login to our admin account. My account name is **admin**. Once logged in let's navigate to the home directory of the admin account and create a hidden folder with specific permissions. Please make sure you are using your own account here.

**cd /home/admin/**

**mkdir .ssh**

**ssh-keygen**

Save the file in **/home/admin/.ssh/id\_rsa**. It created two files. One is the server key that the server will use and the other is the key that we will use. We need to create another file and add our key to it, so the server knows what to use.

**cd .ssh**

**sudo cat id\_rsa.pub > authorized\_keys**

**chmod 700 /home/admin/.ssh**

**chmod 600 /home/admin/.ssh/authorized\_keys**

Let's open the file and see what we have.

**sudo nano authorized\_keys**

Ok close that out and display the id\_rsa key and copy that to our clipboard.

**cat id\_rsa**

Select everything and press **Control+C** to copy it to your clipboard. Open Notepad++ and paste in the key. Make sure to remove the spaces. Then save it as all types. Type in id\_rsa as the file name. Open putty gen and load in the key. Make the comment as **Admin-Server-Key**. Save the key as a private key. Save it in your documents folder.

Open putty and let's test to make sure the key works before locking things down. Create the settings and on the left hand side under Connections -> SSH -> Auth insert the key there. Go back up to Session and save the settings with a name so you no longer have to do that each time you want to connect.

We will try root and it should refuse the key. It does. Let's now try admin. It accepts the key and logs us in. Let's secure the terminal now.

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

**PasswordAuthentication No**

**Control+O**

**Control+X**

**sudo service ssh restart**

Let's try logging in normally. As you can see it fails because we removed the ability to type in passwords. Now if we save the key then we can access the server.

If you have FileZilla, you need to add the key file to the program to work.

And we have secured our server.

# Step 17 – Setup Notifications for Updates to the Server

This will tell you the updates that are pending and can give you a description. Extremely helpful if you are not logging into your server on a day to day bases.

**sudo apt-get install apticron**

**sudo nano /etc/apticron/apticron.conf**

Set email to the email you want it to be sent to.

Save and close. Use the following command to test apticron:

**sudo /usr/sbin/apticron –cron**

You should receive an email, and everything is setup.

# Step 18 – Set Time Zone

We can set the time zone to our location instead of using the default time zone when the server is setup. This helps with automated tasks etc.

**timedatectl list-timezones** to list out the time zones.

**sudo timedatectl set-timezone [timezone]**