DigitalOcean droplet install guide

This install guide describes how I have setup a Nginx powered Ubuntu 14.04 droplet on DigitalOcean for future reference.

The information recorded is a collection of information found around the web, supplemented with my own lines and insights.

Please ping me or send PR for incorrect or obsolete information.

Index

Add users

Change root password after login

```
$ ssh root@<ip address droplet>
$ passwd
```

Update system

```
$ apt-get update
$ apt-get upgrade
$ apt-get autoremove && apt-get autoclean
```

To also upgrade the Ubuntu distribution version run: `apt-get dist-upgrade`

Add users

```
$ adduser <username>
```

Add root privileges for added users

```
$ visudo
```

Now add for every newly added user the following line:

```
<username>      ALL=(ALL:ALL) ALL
```

Now log out root user.

Login with new user

```
$ ssh <username>@<ip address droplet>
```

Create '.bash_profile'

First add a `.bash_profile` file to the user's home directory if not already present (`/root/` or `/home/<username>`):

```
$ cd ~
$ touch .bash_profile
```

Edit '.bash_profile' file

```
$ nano .bash_profile
```

(or edit via Transmit's SFTP)

Add the following 2 lines to the `.bash_profile` file:

```
export LC_ALL=en_US.UTF-8
export LANG=en_US.UTF-8
```

Reload `.bash_profile` (or open a new Terminal window):

```
source .bash_profile
```

Stop forwarding locale from the client

Open the local `ssh_config` file in Sublime Text via Terminal:

```
sublime /etc/ssh_config
```

Now out-comment the following line: `SendEnv LANG LC_*`

Update or add '.gitconfig' file

Upload from the `/git` folder the `gitconfig.sample` config file to user's home directory (`/root/' or` /home/ `) and rename to` .gitconfig`.

Generate and reconfigure the (missing) locales

Login (again) as root user and generate and reconfigure the (missing) locales:

```
locale-gen en_US en_US.UTF-8
dpkg-reconfigure locales
```

Setup SSH keys

If there is no `~/.ssh` directory in your user's home directory already, create one (login with the correct user first!):

```
$ mkdir ~/.ssh
```

If there is no `id_rsa` and `id_rsa.pub` key combo, then generate one:

```
$ ssh-keygen -t rsa
```

Now from your local machine (assuming your public key can be found at `~/.ssh/id_rsa.pub`) enter the following command in Terminal:

```
$ ssh user@hostname/ipadress 'cat >> ~/.ssh/authorized_keys' < ~/.ssh/id_rsa.pub
```

Now if you close the connection, and then attempt to establish a new SSH connection, you should no longer be prompted for a password!

Install Nginx, PHP5-FPM, GD/ImageMagick and Sendmail

To check if a `packagename` was already installed:

```
$ dpkg -s <packagename>
```

```
$ dpkg -l <packagename>
```

Install Nginx

```
$ apt-get install nginx
```

Test run and then (re)start the installation:

```
$ nginx -tt && service nginx restart
```

Install PHP

```
$ apt-get install php5-fpm
```

Then (re)start and check:

```
$ service php5-fpm restart
$ check `dpkg -l php5-fpm
```

Install GD

```
$ apt-get install php5-gd
```

Then (re)start and check:

```
$ dpkg --get-selections | grep php
```

or

```
$ dpkg -l php5-gd
```

Install ImageMagick

```
apt-get install php5-imagick
```

(Re)start and check:

```
$ dpkg --get-selections | grep php
```

or

```
$ dpkg -l php5-imagick
```

Install Sendmail

Optional! Only install SendMail when planning to implement a contact form to your website.

```
apt-get install sendmail
```

(Re)start and check:

```
$ dpkg -l sendmail
```

Now configure sendmail:

```
$ sendmailconfig
```

(basically say 'Y' to all questions)

Edit the `/etc/hosts` file (replace the `127.0.0.1 …` / `127.0.1.1 …` lines with):

```
127.0.0.1   localhost.localdomain   localhost
127.0.1.1   hostname.example.com    hostname/do-droplet-name
ip-address/do-droplet-ip   hostname.example.com   hostname/do-droplet-name
```

- to get hostname/do-droplet-name, type: `hostname`, and…
- to get ip-address/do-droplet-ip, type: `hostname -I`
- now restart hostname: `service hostname restart`
- when the domain is `test.example.com`, the hostname/do-droplet-name `carrot`, and the ip-address/do-droplet-ip `177.55.162.226`, the configuration should look like this:

```
127.0.0.1   localhost.localdomain   localhost
127.0.1.1   test.example.com   carrot
177.55.162.226   test.example.com   carrot
```

More infromation

- When you use multiple domains (e.g. `example.com`, `my.example.com` or `another-example.com`) on the same DigitalOcean droplet, then please let me now *how to configure to send from multiple domains*?
- Read more about Setting the Hostname & Fully Qualified Domain Name (FQDN) on Ubuntu 12.04 or CentOS 6.4
- To apply the new host name without rebooting the system type: `hostname example.com`, and then then check if the correct FQDN is set: `hostname -f`

Configure Nginx

Login (SFTP) as `root` user with Transmit f.i.

Upload config files

Backup the default `nginx.conf` and `mime.types` files by adding a tilde to the file name: `nginx.conf~` and `mime.types~`

Now upload from the `/nginx` folder both the `nginx.conf` and `mime.types` files.

Upload from the `/nginx` folder the config files to: `/etc/nginx`, frist the `hbp5` and (when using Kirby CMS) the `kirby` folders.

Now rename `kirby.conf` file:

```
kirby-example.conf
```

Update the line `fastcgi_pass unix:/var/run/php5-fpm.sock;` in `kirby-example.conf` to link to the correct example socket (to be created later the 'configure php-fm' set below!):

```
fastcgi_pass unix:/var/run/php5-fpm-example.sock;
```

Sites available

Delete the `default` file in the `sites-available` folder and upload (again from the `/nginx` folder) the following files three:

- no-default
- example.com
- ssl.example.com

(Make sure to delete the `default` symlink!)

Rename and edit the contents of the `example.com` and `ssl.example.com` files to follow your server/website setup — among other things, include the correct kirby conf file created in the previous step:

```
include kirby/kirby-example.conf;
```

Sites enabled

Login with `root` user and active the desired virtual hosts (don't forget to angel the no-default site):

```
$ ln -s /etc/nginx/sites-available/example.com /etc/nginx/sites-enabled/example.com
$ ln -s /etc/nginx/sites-available/no-default /etc/nginx/sites-enabled/no-default
```

Test and restart nginx

```
nginx -tt && service nginx restart
```

Setup file permissions (umask)

Umask defines the default set of file and folder permissions.

Umask modes explained

- The default **umask 002** used for normal user. With this mask default directory permissions are `775` and default file permissions are `664`.
- The default **umask for the root user is 022** result into default directory permissions are `755` and default file permissions are `644`.
- For directories, the **base permissions** are (rwxrwxrwx) `0777` and for files they are `0666` (rw-rw-rw).

In short,

- A umask of `022` allows only you to write data, but anyone can read data.
- A umask of `077` is good for a completely private system. No other user can read or write your data if umask is set to 077.
- A umask of `002` is good when you share data with other users in the same group. Members of your group can create and modify data files; those outside your group can read data file, but cannot modify it. Set your umask to `007` to completely exclude users who are not group members.

Set umask mode for user

For my use I need to set the umask mode to `002` …

Both Debian and Ubuntu ship with `pam_umask`. This allows you to configure umask in `/etc/login.defs` and have them apply system-wide, regardless of how a user logs in.

To enable it, you may need to add a line to `/etc/pam.d/common-session` reading:

```
session optional pam_umask.so
```

Or it may already be enabled. Then edit `/etc/login.defs` and change the `UMASK` line to:

```
UMASK           002
```

(the default is `022`)

Note that users may still override umask in their own `~/.profile` or `~/.bashrc` or similar, but (at least on new Debian and Ubuntu installations) there shouldn't be any overriding of umask in `/etc/profile` or `/etc/bash.bashrc`. (If there are, just remove them.)

Set umask mode for PHP-FM

Only when PHP-FM is installed and being used continue…

Add to the to `/etc/init/php-fpm.conf` file, just after line 7 (`stop on runlevel [016]`) the following:

```
umask 002
```

Configure PHP-FM

PHP-FPM creates and manages a pool of php processes, also called workers that receive and server requests to execute php files from the web directory. Now fpm can run multiple separate pools each with its own uid/gid.

On ubuntu, the directory that contains the pool configuration files is:

```
/etc/php5/fpm/pool.d/
```

A file called `www.conf` already exists which can be copied to create more pool configuration files. Each file must end with `.conf` to be recognised as a pool configuration file by php fpm.

Copy conf file

Create a copy of the `www.conf` file and rename it:

```
example.conf
```

Backup (rename) `www.conf` by adding a tilde: `www.conf~`

Edit conf file

Edit the following fields in `example.conf` file:

- Pool name. It is on the top `[www]`. Rename it to `[example]`.
- The user and group fields:

```
user = example
group = example
```

- The socket file name (every pool should have its own separate socket):

```
listen = /var/run/php5-fpm-example.sock
```

Socket files

If not already done in 'configure nginx' step, make sure the `example` site uses the correct socket file to connect to fpm:

Rename the earlier uploaded `/etc/nginx/kirby/kirby.conf` file to:

```
kirby-example.conf
```

Update the line `fastcgi_pass unix:/var/run/php5-fpm.sock;` to link to the correct example socket:

```
fastcgi_pass unix:/var/run/php5-fpm-example.sock;
```

If not already done in 'configure nginx' step, open the earlier uploaded and configured `/etc/nginx/sites-available/(ssl.)example.com` files, and include the correct kirby conf file created in the previous step:

```
include kirby/kirby-example.conf;
```

Now login as root user and restart php-fpm:

```
service php5-fpm restart
```

source

Configure Git

Flavour 1

With bare repositories.
Do you make use of git submodules? See *flavour 2* below!

Install Git

```
$ apt-get install git-core
$ apt-get autoremove && apt-get autoclean
```

Create ‘public’ folders

Create the following folders with `root` user (`-p` makes sure all the directories that don’t exists already are created, not just the last one):

```
$ mkdir -p /usr/share/nginx/www/example.com/public
$ mkdir -p /usr/share/nginx/www/stage.example.com/public
$ mkdir -p /usr/share/nginx/repo/example.git
$ mkdir -p /usr/share/nginx/repo/stage.example.git
```

Update group and user permissions

Now change the group and ownership of the `/public` and `/example.git` folders:

```
$ sudo chown -R example:example /usr/share/nginx/www/example.com/public
$ sudo chown -R example:example /usr/share/nginx/www/stage.example.com/public
$ sudo chown -R example:example /usr/share/nginx/repo/example.git
$ sudo chown -R example:example /usr/share/nginx/repo/stage.example.git
```

Move the `/usr/share/nginx/html/50x.html` file to the newly created `/www` directory: `/usr/share/nginx/www` , and then delete the `/html` directory.

Initialize *bare* git repositories

Login (SSH) with `example` user and initialize the bare Git repositories:

```
$ cd /usr/share/nginx/repo/example.git
$ git init --bare
```

(Repeat for staging domain!)

Git hooks

Make sure to login (either SSH or SFTP) with the correct `example` user when uploading the files, otherwise the file group/owner will be incorrect!

Upload to the `/usr/share/nginx/repo/example.git/hooks` folder of the bare git repository the `post-receive.bare.sample` file, located in the `/git/hooks` folder (make sure to enter the correct virtual host, etc.) and after uploading rename to `post-receive` .

Set permissions of the `post-receive` file to `775` .

Repeat for staging (and other possible) (sub)domain(s).

Sparse checkout

Upload to the `/usr/share/nginx/repo/example.git/info` folder of the bare git repositories the `sparse-checkout.sample` file (set permissions to `664`), located in the `/git/info` folder (make sure to enter the correct path-to-files) and after uploading rename to `sparse-checkout`.

Add remote stage and production repositories

Now add the remote stage and production repositories to your local repository:

```
$ git remote add stage ssh://example@hostname-or-ip/usr/share/nginx/repo/stage.exammple.git`
$ git remote add production ssh://user@hostname-or-ip/usr/share/nginx/repo/example.git
```

Flavour 2

Repositories with submodules. Based on this article.
Do you not make use of git submodules? See *flavour 1* above!

Install Git

```
$ apt-get install git-core
$ apt-get autoremove && apt-get autoclean
```

Create 'public' folders

Create the following folders with `root` user (`-p` makes sure all the directories that don't exists already are created, not just the last one):

```
$ mkdir -p /usr/share/nginx/www/example.com/public
$ mkdir -p /usr/share/nginx/www/stage.example.com/public
```

Update group and user permissions

Change group and ownership of the `/public` folders:

```
$ sudo chown -R example:example /usr/share/nginx/www/example.com/public
$ sudo chown -R example:example /usr/share/nginx/www/stage.example.com/public
```

Move the `/usr/share/nginx/html/50x.html` file to the newly created `/www` directory: `/usr/share/nginx/www`, and then delete the `/html` directory.

Initialize git repositories

Login (SSH) with `example` user and initialize the Git repositories:

```
$ cd /usr/share/nginx/www/example.com/public
$ git init
```

(Repeat for staging domain!)

Git hooks

Make sure to login (either SSH or SFTP) with the correct `example` user when uploading the file, otherwise the file group/owner will be incorrect!

Upload to the `/usr/share/nginx/www/example.com/.git/hooks` folder the `post-receive.submodules.sample` file, located in the `/git/hooks` folder (make sure to enter the correct virtual host, etc.) and after uploading rename to `post-receive`.

Set permissions of the `post-receive` file to `775`.

Repeat for staging (and other (sub)domains).

Sparse checkout

Now add to the `/usr/share/nginx/www/example.com/.git/info` folder the `sparse-checkout.sample` file (set permissions to `664`), located in the `/git/info` folder (make sure to enter the correct path-to-files) and after uploading rename to `sparse-checkout`.

Add remote stage and production repositories

Now add the remote stage and production repositories to your local repository:

```
$ git remote add stage ssh://example@hostname-or-ip/usr/share/nginx/www/stage.exammple.com/public
$ git remote add production ssh://user@hostname-or-ip/usr/share/nginx/www/example.com/public
```

Setup Dropbox sync

Login (SHH) with the special `dropbox` user (add user if not done already).

Make sure to be in the dropbox's home directory:

```
cd ~
```

Install Dropbox client

Download and install dropbox client:

```
$ Stable 32-bit: `wget -O dropbox.tar.gz "http://www.dropbox.com/download/?plat=lnx.x86"
```

or when running Ubuntu 64-bit:

```
$ Stable 64-bit: `wget -O dropbox.tar.gz "http://www.dropbox.com/download/?plat=lnx.x86_64"
```

Sanity check to make sure we're not going to clog our home directory:

```
$ tar -tzf dropbox.tar.gz
```

Now extract:

```
$ tar -xvzf dropbox.tar.gz
```

Run dropboxd

```
$ ~/.dropbox-dist/dropboxd
```

You should see output like this:

```
This client is not linked to any account... Please visit https://www.dropbox.com/cli_link?host_id=7d44a557aa
```

Go to the URL given, and after success, dropboxd will create a `~/Dropbox` folder and start synchronizing it after this step.

Dropbox CLI

It is recommended to download the official Dropbox CLI to start the dropbox daemon (as an unprivileged user) and get its status:

```
$ mkdir -p ~/bin
$ wget -O ~/bin/dropbox.py "http://www.dropbox.com/download?dl=packages/dropbox.py"
$ chmod 755 ~/bin/dropbox.py
$ ~/bin/dropbox.py help
```

Change the too restrictive default Dropbox folder permissions:

```
$ chmod 755 ~/Dropbox
```

Create symlink to the `dropbox` file in the `~/.dropbox-dist/` folder (necessary for folowing steps):

```
$ ln -s ~/.dropbox-dist/dropbox-lnx.x86-2.10.28/dropbox ~/.dropbox-dist/dropbox
```

To run dropbox on system startup, login (SSH) as `root` user and then create: `touch /etc/init.d/dropbox`

Edit the newly created file and add the contents of the `/dropbox/dropbox.sample` file to `/et/init.d/dropbox`.

Make the script is executable and add it to the system startup:

```
$ chmod +x /etc/init.d/dropbox
$ update-rc.d dropbox defaults
```

Now control the Dropbox client like any other Ubuntu service:

```
$ service dropbox start|stop|reload|force-reload|restart|status
```

Symlink folders

Still logged in as `root` user, add symbolic links to the `public` content folders of the example.com site:

```
$ example.com  `ln -s /home/dropbox/Dropbox/example.com/content/ /usr/share/nginx/www/example.com/public/con
$ stage.example.com  `ln -s /home/dropbox/Dropbox/example.com/content/ /usr/share/nginx/www/stage.example.co
```

Notes

- If something went wrong during the install you can try again by deleting all the Dropbox files in the home directory: `rm -rf .dropbox* Dropbox`. And start again by downloading the files you need.
- If you want to change the Dropbox account it is linked to, unlink it from the first account, then kill the running dropbox process, start it up again (with `~/.dropbox-dist/dropboxd &`) and obtain the new host_id with `dbreadconfig.py`. If you don't restart the dropbox client, it will give the same host_id (which for some reason causes it to be unable to change the account it is linked to).

Setup BitTorrent sync

TBA

Set permissions to files and (sub)folders all at one time

To change all the directories to 755 (-rwxr-xr-x):

```
$ find /usr/share/nginx/www/example.com/public/ -type d -exec chmod 775 {} \;
```

$ To change all the files to 644 (-rw-r–r–):

```
find /usr/share/nginx/www/example.com/public/ -type f -exec chmod 664 {} \;
```

Check for AES-NI support

To use AES NI you need to load the aesni_intel kernel module:

```
$ /sbin/modinfo aesni_intel
filename:       /lib/modules/3.0.0-13-generic/kernel/arch/x86/crypto/aesni-intel.ko
alias:          aes
license:        GPL
description:    Rijndael (AES) Cipher Algorithm, Intel AES-NI instructions optimized
srcversion:     61A51F44F192D7CE0FBA795
depends:        cryptd,aes-x86_64
vermagic:       3.0.0-13-generic SMP mod_unload modversions
```

To see if your CPU supports AES NI check the output of "cat /proc/cpuinfo l grep aes”:

```
$ cat /proc/cpuinfo | grep aes | wc -l
4
```

To check whether or not AES NI is enabled check the contents of /proc/crypto:

```
$ grep module /proc/crypto | sort -u
module       : aesni_intel
module       : aes_x86_64
module       : arc4
module       : kernel
```

To see if OpenSSL supports AES-NI run openssl engine:

```
$ openssl engine
(aesni) Intel AES-NI engine
(dynamic) Dynamic engine loading support
```