DEPLOYING LARAVEL AND NUXT ON LINUX SERVER – Start to FInish

Buy a server on Digital Ocean

1. Go to Digital Ocean https://m.do.co/c/5b428a74fc09 to sign up. **Use this link to get $100 in free credit.**
2. Create a new Project
3. Create a new Droplet and OS
   a. Click on "**Get started with a Droplet**".
   b. Choose Ubuntu LTS
   c. Select plan ($5/mo is ok)
   d. Choose a datacenter closest to you

Setup SSH

Create an SSH Key on your local (development) computer

# make the .ssh directory if it doesn't already exist (-p)
```
>> mkdir -p ~/.ssh
```

# go to your ssh folder
```
>> cd ~/.ssh
```

# list all the files in this location
```
>> ls
```

# Generate the ssh key
```
>> ssh-keygen
```

The default key name is **id_rsa**, so if you already have one and don't want to use that one, then use a different name otherwise it will conflict. I will use `<< designhouse >>.`

You can enter a passphrase for your key, but I will leave it without a passphrase.

# copy the public key to Digital Ocean
```
>> cat <<name of key>>.pub
```

# SSH into the server as ROOT from your development computer (-i stands for Identity File)
```
>> ssh -i shipping root@ip-address
```

Now we're in. But it is a pain to have to enter then name of the identity file each time. So, we will add the file name to the Keychain so that next time it will know where to look.

# 1. make sure that keychain is active.
>># Check if config file exists
```
>> ls ~/.ssh
```

# if not exists, create a config file
```
>> touch ~/.ssh/config
```

# open the config file
```
>> vi ~/.ssh/config
```

# add the following to the vile
```
Host *
 AddKeysToAgent yes
 UseKeychain yes
 IdentityFile ~/.ssh/id_rsa
```

This will allow the SSH file to be automatically added to the keychain when we successfully login to the server

Buy Domain name from NameCheap

Go to Namecheap and buy a domain (or to any provider of your choice)
I am using designhouse.party

Add DNS Records to Digital Ocean

>> An A-Record maps a Domain name to an IP Address
>> A CNAME record maps a canonical name to another canonical name. Eg. mapping a subdomain to the main domain or to another subdomain

```
>> dig google.com
>> dig mail.google.com
```

# Add 2 A Records in DO
```
>> @ which is the root domain.
>> www which is the www.domain.com
```

Add NameServers

The nameserver is like a Phonebook. We need to copy the nameservers from Digital Ocean to Namecheap (or the domain provider you chose)

It may take up to 48 hours for the nameservers to propagate, so we will let that happen while we continue to provision our server

Setup our Server

- ➢ Update the OS Software
- ➢ Create a new user
- ➢ Make the new user a Super user
- ➢ Enable SSH login for the new user
- ➢ Disable ROOT Login

# Update Software
```
>> apt update
```

# Upgrade software (to make sure we have the latest of everything)
```
>> apt upgrade
```

# add new user and enter a password
```
>> adduser gabs
```

# Add the user to the sudo group
```
>> usermod -aG sudo gabs
```

# Switch to the new user
```
>> su gabs
```

# Check user's sudo access
```
>> sudo cat /var/log/auth.log
```

# SETUP PERMISSIONS AND SSH FOR NEW USER

# Go to the user's home directory
```
>> cd ~
```

# Create a new .ssh directory if not already exists
```
>> mkdir -p ~/.ssh
>> ls -a  #-a allows you to see the hidden files and folders
```

# Create an authorized_keys file and pasge the PUBLIC key
```
>> vi ~/.ssh/authorized_keys
```

# Copy the public key from the local computer into the authorized_keys file

# Exit and ssh again as the new user
```
>> ssh gabs@ip-address
```

# DISABLE ROOT LOGIN
```
>> sudo vi /etc/ssh/sshd_config
# set PermitRootLogin to no
```

# Restart the sshd service
```
>> sudo service sshd restart
```

# Exit from the server and try to login again as ROOT, which should fail. Login again as gabs

SETUP WEB SERVER

SETUP FIREWALL

# Enable Firewall (ufw – uncomplicated firewall) so that only connections to certain services are allowed. Once enabled, some applications can register their profiles to this firewall when installed.

# See the list of applications that currently have a profile on UFW. This lists OpenSSH, which # is the application that allows us to ssh into the server now
```
>> sudo ufw app list
```

# Make sure that the firewall allows SSH connections
```
>> sudo ufw allow OpenSSH
```

# Then Enable the firewall
```
>> sudo ufw enable
```

# Check the firewall status
```
>> sudo ufw status
```

# So now, Firewall is blocking all connections except SSH. So if you install and configure any additional services, you will need to adjust the firewall to allow connections for these services as well

INSTALL A LEMP STACK (Linux, Nginx, MySQL, PHP)

- Install NginX
- Install MySQL
- Install PHP

A. Install NginX

```
>> sudo apt update
>> sudo apt install nginx
```

# Allow connections to NginX in the firewall

```
>> sudo ufw allow ‘Nginx HTTP’
>> sudo ufw status
```

# Copy the IP Address from DO and test it in the server. This should show the Nginx page

B. Install MySQL

# Install MySQL Server

```
>> sudo apt install mysql-server
```

# The server comes with a script that allows us to change some insecure defaults

```
>> sudo mysql_secure_installation
```

### Answer NO for VALIDATE PASSWORD PLUGIN

### Enter a root password
### Press YES and ENTER to remove anonymous users from the server and for all other options

# Launch mysql and enable root password login (by default, it used socket authentication – auth_socket - , which is ok but may cause issues if you want to login via password services like PHPMyAdmin)

```
>> sudo mysql
```

# Check which authentication method each MySQL user account is using

```
>> SELECT user, plugin, host from mysql.user;
```

# Change root user to authenticate with password instead of auth_socket

```
>> ALTER USER ‘root’@’localhost’ IDENTIFIED WITH
mysql_native_password BY ‘password’;
```

# Flush priviledges so that the server can reload the grants table and apply your changes

```
>> FLUSH PRIVILEGES
```

# Check he query again

```
>> SELECT user, plugin, host from mysql.user;
>> exit
```

# Because we have changed the root login to use password, we can no longer access mysql using

```
>> sudo mysql # will no longer work
>> mysql -u root -p # will now work
```

C. Install PHP and configure Nginx to use the PHP Processor

Unlike other webservers like Apache, Nginx does not natively contain a PHP processor. So we need to install the fastCGI process manager (or php-fpm) and then tell Nginx to forward all PHP requests to this process manager for processing.

# Add the Ubuntu universe repository (contains several free software from the Ubuntu community)

```
>> sudo add-apt-repository universe
```

# Install php-fpm and php-mysql which allows PHP to talk to the database

```
>> sudo apt install php-fpm php-mysql
```

# Create a Server Block for NginX (similar to VirtualHost in Apache Server). This is to tell NginX to use the PHP Processor for dynamic content. We will call our example.com but you can call it anything. By using a different serverblock file and not the original (default), we can easily restore the default configuration if necessary.

```
>> sudo vim /etc/nginx/sites-available/example.com
```

# paste the following

```
server {
     # Port that nginx should listen on
     listen 80;

     # document root
     root /var/www/html;

     # your domain or ip address
     server_name example.com;

     # priority file extensions
     index index.php index.html index.htm index.nginx-
debian.html;

     # check the existence of files matching a provided url,
     # and forward to 404 if not found
     location / {
          try_files $uri $uri/ =404;
     }
     # handles php processing
     location ~ \.php$ {
          include snippets/fastcgi-php.conf;
          fastcgi_pass unix:/var/run/php/php7.2-fpm.sock;
     }
     # prevent any .htaccess files from being served to the
user
     location ~ /\.ht {
          deny all;
     }
}
```

# create a symlink to point the new file to the sites-enabled

```
>> sudo ln -s /etc/nginx/sites-available/example.com
/etc/nginx/sites-enabled/
```

# then unlink the default configuration from the sites-enabled
```
>> sudo unlink /etc/nginx/sites-enabled/default
```

# check for syntax errors in the nginx config
```
>> sudo nginx -t
>> sudo systemctl reload nginx
```

# create a info.php to test
```
>> sudo vim /var/www/html/info.php
<?php phpinfo();
```

# then visit `IP_ADDRESS/info.php`
# then we can remove the `info.php file`

END OF LEMP STACK CONFIG

Install Composer for Dependency management

# Install the dependencies
```
>> sudo apt install curl php-cli php-mbstring git unzip
```

# Download composer, verify that the package is not corrupt and then install it
```
>> cd ~ # go to your home directory
>> curl -sS https://getcomposer.org/installer -o composer-
setup.php
```

# Install composer globally
```
>> sudo php composer-setup.php --install-dir=/usr/local/bin --
filename=composer
```

# Test composer
```
>> composer
```

Install other PHP modules required by Laravel

```
>> sudo apt install php-mbstring php-xml php-bcmath php7.2-gd
```

Create a MySQL Database

```
>> mysql -u root -p
>> CREATE DATABASE <designhouse>;
```

# Create a new user (shipping_user) and grant all permissions on the database to the user
```
>> GRANT ALL ON shipping.* TO 'shipping_user'@'localhost'
IDENTIFIED BY 'password';
>> exit
```

# Test if the new user has access to the database
```
>> mysql -u shipping_user -p
>> show databases; #NB this user only has access to the
database assigned to them
>> exit
```

Pull Laravel app from Github

# Go to home directory
```
>> cd ~
```

# Genetate SSH key and add to GitHub
```
>> ssh-keygen
>> cat ~/.ssh/id_rsa.pub
# copy the key over to Github
```

# clone project from github
```
>> git clone git@github.com:nfunwigabga/designhouse-api.git
api
>> cd api
>> composer install
>> cp .env.example .env
>> vim .env
>> update credentials
>> php artisan key:generate
>> php artisan jwt:secret
>> php artisan migrate
```

# Check which user is running PHP
```
>> ps aux | grep php
```

# Give the web server user ownership and write access our folder
```
>> sudo chown -R www-data:www-data .
>> sudo chown -R www-data:www-data .
```

# Also add your user account to the www group
```
>> sudo usermod -aG www-data gabs
```

# change permissions to storage folder
```
>> sudo chmod -R 775 storage/
```

# Link storage folder to Home directory
```
>> php artisan storage:link
```

# Create a new Nginx Server Block for our new site, similar to what we created in example.com before
```
>> sudo vim /etc/nginx/sites-available/designhouse
```

# create symlink
```
>> sudo ln -s /etc/nginx/sites-available/shipping
/etc/nginx/sites-enabled/
>> sudo nginx -t
>> sudo systemctl reload nginx
```

# visit the IP Address

Updated Config

```
server {
    listen 80;
    server_name designhouse.party;
    root /home/gabs/api/public;
    add_header X-Frame-Options "SAMEORIGIN";
    add_header X-XSS-Protection "1; mode=block";
    add_header X-Content-Type-Options "nosniff";

    index index.html index.htm index.php;

    charset utf-8;

    location / {
        try_files $uri $uri/ /index.php?$query_string;
    }

     # Serve static files directly
     location ~* ^/storage/(.*)\.(jpg|jpeg|gif|bmp|png|ico)$ {
          access_log off;
     }
```

```
    error_page 404 /index.php;

    location ~ \.php$ {
        fastcgi_pass unix:/var/run/php/php7.2-fpm.sock;
        fastcgi_index index.php;
        fastcgi_param SCRIPT_FILENAME
$realpath_root$fastcgi_script_name;
        include fastcgi_params;
    }

    location ~ /\.(?!well-known).* {
        deny all;
    }
}
```

Setup NPM

Install Nodejs using PPA (Personal Package Archive)s and npm

```
>> cd ~
>> curl -sL https://deb.nodesource.com/setup_12.x -o
nodesource_setup.sh
```

# Add the PPA to your configuration and update the local cache

```
>> sudo bash nodesource_setup.sh
```

# Now install nodejs and npm

```
>> sudo apt install nodejs
>> node -v
>> npm -v
```

Clone the Nuxt App from GitHub

# Clone the app

```
>> cd ~
>> git clone git@github.com:nfunwigabga/designhouse-client.git
client
```

```
>> cd client
>> npm install
```

# make .env file
```
>> vi .env
```

# Content. We will change the url later with a domain name
```
BASE_URL=http://designhouse.party
API_URL=http:// designhouse.com/api
```

# Build the app in production
```
>> npm run build
```

# Try to run the app. You will see it is running on Port 3000. But we cannot access this.
```
>> npm run start
```

Use ProxyPass

# We need to make some changes to our Nginx config to be able to access the app. We will set up a reverse proxy to forward all requests coming to port 80 (which the the port through which our website is accessible) to port 3000 (which is the port for our Nuxt App.

```
>> sudo vim /etc/nginx/sites-available/shipping
```

```
# forward api requests to PHP
location /api{
        try_files $uri $uri/ /index.php?$query_string;
 }

# replace the / directive with the following
location / {
     proxy_pass http://127.0.0.1:3000/;
     proxy_set_header Host $host;
     proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header X-Forwarded-Proto $scheme;
}
```

```
>> sudo nginx -t
>> sudo systemctl reload nginx
```

So our API is now only accessible through /api, while other routes will be redirected to the Nuxt config. At the moment, our Nuxt app is not working, but we can manually run it and see what happens:
`>> npm run start` # Then visit the site

Process Manager (pm2)

# Install pm2
```
>> sudo npm i -g pm2
>> pm2 list
>> pm2 start npm --name “designhouse” – start
```

# Ensure that pm2 starts up automatically after the server restarts
```
>> pm2 startup
>> copy the setup and paste
>> pm2 save
```

# Go to Digital Ocean and Turn Off the server
# try to access the web page again

Configure SSL Certificate

We will add a Free SSL Certificate to our domain using CertBot. Certbot is a wrapper around LetsEncrypt (A free Certificate Authority). Certbot was created by a project called Electronic Frontier Foundation (EFF) (https://certbot.eff.org/).

>> Go to https://certbot.eff.org
>> Select the Web server (Nginx) and OS (Ubuntu 18).
>> We can just follow the instructions on the page to install SSL

# After installation, add HTTPS to ufw Firewall

```
>> sudo ufw status
>> sudo ufw allow 'Nginx Full'
>> sudo ufw delete allow 'Nginx HTTP'
>> sudo systemctl reload nginx
```

# Now visit your website