==

SECTION 1 **Note: anything in bold is a command you must run**. Anythin in red is sujective to your own path/ file names/ ip.

==

NEVER EVER RUN SUDO INSIDE AN ACTIVE VIRTUAL ENVIORNMENT. SUDO IS FOR SYSTEM WIDE COMMANDS ONLY.

Stage 0 - Prepare your project

NOTE: make sure you don't have any weird packages inside your project. If you do, add it to the requirements file. Gunicorn will crash if you don't have it installed. Look through your installed apps in settings and just double check.

1) Create a requirements.txt file at the project level inside pycharm with these contents.

```
Django==1.8.4
django-bootstrap-form==3.2
django-debug-toolbar==1.3.2
django-disqus==0.5
django-forms-bootstrap==3.0.1
Pillow==3.0.0
python-dateutil==2.4.2
requests==2.8.0
sqlparse==0.1.16
stripe==1.24.1
virtualenv==13.1.2
wheel==0.24.0
mysql-python
```

2) Create a .gitignore file at project level with these contents

```
*.pyc
*.wp*
*.sqlite3
local_settings.py
```

3) Duplicate your settings file. Call the second file local_settings.py

Do not touch this local setting file now. This is where all your local settings are configured.

4) Open up the settings.py file

Change these items exactly as I tell you

```
ALLOWED_HOSTS = ['*']

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'setadbname',
        'USER': 'setausername',
        'PASSWORD': 'setapassword',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}
```

5) Cut out the end of the settings.py file that has all the static and media and replace it with this. Its basically the exact same bar a few lines. This just tells django to use the local settings if on a local machine or use the real settings if on a linux box!!!

```
STATIC_URL = '/static/'
STATIC_ROOT = os.path.join(BASE_DIR, "staticfiles")
STATICFILES_DIRS = ( os.path.join(BASE_DIR, "static"),)

MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
MEDIA_URL = '/media/'

try:
    from local_settings import *
except:
    pass
```

6) Go to your global urls.py file and import the following lines

```
import settings
from django.conf.urls.static import static
```

attach this to the very end outside the url patterns brackets

```
+ static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT) +
  static(settings.STATIC_URL,document_root=settings.STATIC_ROOT)
```

If you are stuck, go to my github and look at how I have done it. Its really very very simple. The project is in the repos fleeky_notif

7) Run your project on your local machine and make sure it works first!!!!

===

<u>Stage1 - Get your project onto github.</u>

Go to your github and setup a new repository

Go to your terminal and navigate to the root level of your projects folder.

git init
git add .
git commit -m "myfirstcommit"
git remote add origin   addressofyourrepository
git push -u origin master

Great. Now your project is on github.

8) MAKE SURE THAT .GITIGNORE WORKED. IT SHOULD HAVE DELETED ALL THE .pyc files. If not, go through your github project and manually delete them.

===

<u>Stage2 - Linode initiate</u>

Sign up to Linode
Deploy a new Ubuntu 15.04 image onto that Linode and use the default settings.
Set your password

Boot the Linode
go to the remote access tab and copy and paste the ssh@yourserversip

==

<u>Stage 3 - Prepare your linux server</u>

ssh your Linode using ssh@yourserversip

Update linux:

apt-get update && apt-get upgrade

Update apt-get:

apt-get update (duh)

Install the system wide packages by using the sudo command. DO NOT SUDO INSTALL GUNICORN HERE:

sudo apt-get install python-pip python-virtualenv python-dev supervisor nginx git
sudo pip install setproctitle

==

<u>Stage 4 - Setup SQl</u>

Install:
sudo apt-get install mysql-server

Login:
mysql -u root -p

Note!! Must use the semi colons at the end of the sql commands! Create the database. NEEDS TO BE SAME NAME AS IN SETTINGS.PY:

create database yourdatabasename;

Create user. NEEDS TO BE SAME USER AS IN SETTINGS.PY

create user 'yourname'@'localhost' identified by 'password';

N.B - dont forget the full stop below, its hard to see.

grant all on yourdatabasename.* to 'yourusername';

grant all privileges on * . * to ‘yourusername’@‘locahost’;

exit

==

<u>Stage 5 - Setup virtual environment and clone git folder onto linux machine</u>

Login to your linux box

cd /
sudo mkdir webapps
cd webapps
virtualenv —no-site-packages myvirtualenv (note there are two dashes before no)

cd myvirtualenv

source bin/activate

git clone copyyourgitfolderaddress

cd thatgitproject

Install the MYSQL dependencies. This is how django communicates with mysql. You can link django with any database you want depending on the dependency.

sudo apt-get -y install libjpeg-dev zlib1g-dev libmysqlclient-dev

pip install -r requirements.txt

python manage.py collectstatic

Note, it will very likely crash here. Don’t panic, this is ok. Its probably because you don’t have the right packages installed. Go through your list and hunt down what you are missing or what you need to get rid off. This part takes patience.

If collectstatic worked, great! Moving on.

python manage.py makemigrations
python manage.py migrate

python manage.py createsuperuser

===

Stage 6- Install gunicorn and bind to your server ip

Finally, lets run a gunicorn test and see if we can serve the django project from django itself. Remember, this is an improper method as nginx should really be serving the static files, but its a good test to know we are on the right track.

MAKE SURE YOU ARE INSIDE YOUR VIRTUAL ENV WITH IT ACTIVATED. N.B

pip install gunicorn

gunicorn nameofsettingsfolder.wsgi:application - - bind yourserversip

Well done!

===

END OF SECTION 1. SECTION 2 ON NEXT PAGE.

===

===

SECTION 2: **Note: anything in bold is a command you must run**. Anythin in red is sujective to your own path/ file names/ ip.

===

NEVER EVER RUN SUDO INSIDE AN ACTIVE VIRTUAL ENVIORNMENT. SUDO IS FOR SYSTEM WIDE COMMANDS ONLY.

Stage 7 - setup a sub user and add the webapps as a group.

The entire purpose of this part is to allow other users to sign into your llinux box and have admin access to the webapps folder. It will alllow them access to the webapps folder but not root access to the entire machine.

Deactivate your virtualenv

add webapps to the system's groups

sudo groupadd --system webapps

add a user for that group

sudo useradd --system --gid webapps --shell /bin/bash --home /webapps/virtualenvname hello

Note, hello is the name of the user I chose. Choose whatever you want, just remember it. (duh)

Give that user permission to everything in that folder

sudo chown -R hello:webapps /webapps/virtualenv

logout

ssh back in

Run this little command

sudo usermod -a -G webapps `whoami`

==============================

Stage 8 - Make and configure a gunciorn file

When you are at the same directory inside your virtual machine as bin

nano bin/gunicorn_start

Type the following in and obviously configure for your own directory names.

———————————————————————————————————-

```
#!/bin/bash

NAME="nameofyourproject"
DJANGODIR=/webapps/virtualmachinename/projectname
SOCKFILE=/webapps/virtualmachine/run/gunicorn.sock   #this is the socket that connects to Nginx!
USER=hello
GROUP=webapps
NUM_WORKERS=3
DJANGO_SETTINGS_MODULE=nameofsettingsfolder.settings
DJANGO_WSGI_MODULE=nameofsettingsfolder.wsgi

echo "Starting $NAME as `whoami`"


cd $DJANGODIR
source ../bin/activate
export DJANGO_SETTINGS_MODULE=$DJANGO_SETTINGS_MODULE
export PYTHONPATH=$DJANGODIR:$PYTHONPATH

RUNDIR=$(dirname $SOCKFILE)
test -d $RUNDIR || mkdir -p $RUNDIR

exec ../bin/gunicorn ${DJANGO_WSGI_MODULE}:application \
  --name $NAME \
  --workers $NUM_WORKERS \
  --user=$USER --group=$GROUP \
  --bind=unix:$SOCKFILE \
  --log-level=debug \
  —log-file=-
```

———————————————————————————————————-

save the file

ctrl X and **YES**

deactivate the virtual machine.

Make the file executable :

sudo chmod u+x bin/gunicorn_start

Why? We dont want to do a default gunicorn launch that we did in the previous section. We want to launch gunicorn with our special config file.

Run the file!

bin/gunicorn_start

N.B. This is very important. Gunicorn should start up nicely with no errors. If you visit your servers ip on port 8000, nothing will appear. This is good. This is what we want because it means gunicorn is searching for files being served by nginx through the socket we declared in our config file. SINCE WE HAVE NOT YET CONFIURED NGINX, this will obviously not work yet. Thus, I would be worried if your website is somehow showing up at this stage.

Try the servers ip address without port 8000. It should still say welcome to nginx. This is what we want.

Ctrl + C to shut down gunicorn.

==

Stage 9 - configure Nginx and connect it to gunicorn!

READ AND UNDERSTAND THIS OR PREPARE TO SUFFER THE WRATH OF LINUX

Let's establish something before we start this part and give a brief run down about how it works.

Nginx has two different folders. Sites available and sites enabled. Both folders have default configuration files. What we will do is create our own custom config file in the sites available folder then once that is saved we will run a command to delete the default file from the sites enabled folder. DO NOT REMOVE THE DEFAULT CONFIG FILE FROM SITES AVAILABLE. ONLY FROM SITES ENABLED.

Finally we will run a linking command that will bridge the sites available folder to the sites enabled folder.

All we need to do then is create the folder for nginx to log its errors in, restart nginx then run the bin/gunicorn_start. When you then go to your servers ip address or domain name, you don't need to use that ugly port number. It servers the files alot faster and in a proper fashion using nginx!

Let's get started.

Navigate to sites available folder

cd /etc/nginx/sites-available

Create the file with the name of your profject
nano nameofproject.conf

Copy the black writing verbatim, the red is subjective to you. (duh)

———————————————————————————————————-

```
upstream serverip or domain{

  server unix:/webapps/nameofvirtualenv/run/gunicorn.sock fail_timeout=0;
}

server {

    listen   80;
    server_name serverip or domain;

    client_max_body_size 4G;

    access_log /webapps/nameofvirtualenv/logs/nginx-access.log;
    error_log /webapps/nameofvirtualenv/logs/nginx-error.log;

#The name staticfiles is very important. remember, collectstatic puts it in staticfiles!

    location /static/ {
        alias   /webapps/nameofvirtualenv/nameofproject/staticfiles/;
    }

    location /media/ {
        alias   /webapps/nameofvirtualenv/nameofproject/media/;
    }

    location / {
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;


        proxy_set_header Host $http_host;


        proxy_redirect off;

        if (!-f $request_filename) {
            proxy_pass http://serverip or domain;
            break;
        }
    }

#you can create custom html error pages if you want
    error_page 500 502 503 504 /500.html;
    location = /500.html {
        root /webapps/nameofvirtualenv/nameofproject/staticfiles/;
    }
}
```

—————————————————————————————————————-

Now, like I said in that really important paragraph at the start of this section, the one that you probably didn't read because you're lazy, lets remove that default file from the SITES ENABLED FOLDER.

sudo rm /etc/nginx/sites-enabled/default

NB!!! NOTE: THIS IS IMPORTANT. Nginx is very volatile and will crash with the smallest things. DO NOT FORGET TO MAKE THE LOGS FOLDER.

Let’s go to the same directory that we specified in the file above so our error logs can get stored.This should be the same folder as the bin!

mkdir logs

N.B !! We need to create as symbolic link between sites available and sites enabled. Run this command to achieve this.

ln -s /etc/nginx/sites-available/nameofconffile.conf /etc/nginx/sites-enabled/nameofconffile.conf

Finally, lets restart nginx.

WITH VIRTUALENV DEACTIVATED:

sudo service nginx restart

If there are no errors, congrats! Its time to launch our gunicorn! Navigate to the folder containing bin. Run our executable file.

bin/gunicorn_start

Your website should now be available from your server ip address, not on port 8000!!!!!

==

Stage 10 (optional) - buy a domain name and reverse dns it to linode

Buy a domain name on godaddy.com (duh)

On godaddy, go to the settings of that domain once purchased.

Find the manage named servers tab and select into it.

Remove any named servers currently in there and replace it with these.

NS1.LINODE.COM
NS2.LINODE.COM
NS3.LINODE.COM
NS4.LINODE.COM
NS5.LINODE.COM

Once this is done, itll take in the range of 4 to 24 hours to activate.

Once complete, go to your linode account.

Click the DNS manager tab and click add a domain zone.

Add the domain name you own and the email address used to purchase it.

Finally,

Go to the remote access tab and click reverse dns. This will reverse your sever ip to your domain name.

Enter your domain name (duh)

You may need to change your Nginx file to the name of your new domain name.

==

<u>Stage 11 (OPTIONAL) - keep your gunicorn running if the server crashes or reboots with supervisor</u>

This is not entirely nessacary but will help you sleep at night if you are worried of a server reboot. Supervisor will automatically relaucnh bin/gunicorn_start

Its really easy.

deativate your virtual env.

Run a system wide command to configure supervisor.

sudo nano /etc/supervisor/conf.d/paid_site.conf

————————————————————————————————————-

```
[program:nameofproject]
command = /webapps/virtualenv/bin/gunicorn_start
user= youruser

stdout_logfile = /webapps/virtulenv/logs/gunicorn_supervisor.log

redirect_stderr = true

environment = LANG=en_US.UTF-8,LC_ALL=en_US.UTF-8
```

————————————————————————————————————-

Finally I will show you a few commands that will get supervisor up and running.

supervisorctl reread

It should tell you if your project is available.

This next command initiates the supervisor process

supervisorctl update

Should return: added process group

Lets run it.

```
sudo supervisorctl start nameofprojectspecified
sudo supervisorctl stop nameofprojectspecified
sudo supervisorctl restart nameofprojectspecified
```

FINISHED