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

HELPFUL VI COMMANDS

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

* Use 'shift-:' and 'wq' enter to close and save the settings.py changes
  + Or ‘shift+zz’

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

OTHER NOTES:

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

To Check Status of NGINX:

sudo systemctl status nginx

To Stop Nginx, run the following command:

sudo systemctl stop nginx

To Start Nginx:

sudo systemctl start nginx

To Gracefully Restart Nginx after changing the configuration:

sudo systemctl reload nginx

To Force Restart Nginx:

sudo systemctl restart nginx

To find process for gunicorn run

sudo lsof -i:8000

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

HELPFUL LINKS

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

[SHOBI SETUP EC2](https://www.youtube.com/watch?v=u0oEIqQV_-E&list=PLX4uXM5lVU53JbQ_1ijxpU0qZIOrJOG--&index=1)

[POSTGRES on EC2](https://adeshg7.medium.com/deploy-django-postgresql-on-ec2-using-apache2-from-scratch-42dc8e6682c1)

[DJANGO on EC2](https://medium.com/saarthi-ai/ec2apachedjango-838e3f6014ab)

[HANDLING STATIC FILES](https://www.youtube.com/watch?v=_TBw7ALJp0Y&list=PLX4uXM5lVU53JbQ_1ijxpU0qZIOrJOG--&index=2)

[PSYCOPG2 and Other Commands](https://www.codevoila.com/post/2/python3-connect-postgresql-with-psycopg2-on-ubuntu)

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

SETUP AWS EC2 INSTANCE:

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

Open your AWS account.

Choose EC2 to initialize a Ubuntu Server and upgrade Operating System.

Click on "Launch Instance”

Choose "Ubuntu Server LTS (HVM), SSD Volume Type"

Choose "General purpose" with one of the "Free for analysis" Type,

Then Click "Review and Launch"

Click on "Launch"

Create "Key Pair" or select an existing one (steps to create one are below)

Type in Key Pair Name and "Launch Instance"

Download to your PC

Open a terminal on your PC and go to your directory with the downloaded .pem file

Make the file read only:`

chmod 400 xxxxxxxxxxxx.pem

Go to EC2 Console

Click on Running Instances

Click on the Public DNS (IPv4) is the Domain Name

Rename the Instance Name

Based on the client that the app is for (e.g. proficere\_zl for Zen Life)

Go to EC2 Console

Click on lefthand "Network & Security" option "Security Groups"

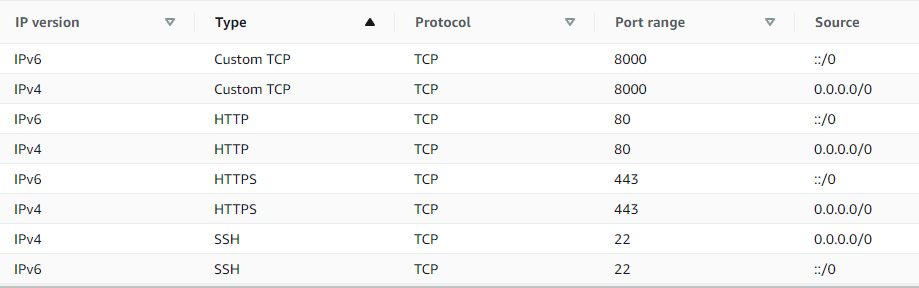
Edit an existing Security Group and click "Assign Security Groups"

"Edit" Security Group "Inbound" tab option to

"Add Rule"

Type = HTTP Port = 80 Source = Anywhere

Type = Customer TCP Port = 8000



(These are the basic security rules that you will need)

Connect to the AWS Server using Key Pair

Right click on the Instance and select "Connect"

You will see an example to use

ssh -i "YourKeyPairNameHer.PEM" ubuntuxxxxxxxxxx.com

Go to the Bash Terminal on your PC in MS-Visual Studio and run the example above from AWS

Update existing AWS software:

sudo apt-get update

sudo apt-get upgrade -y

Verify Python3 exists/has version

python3 --version

Install Python3 VirtualENV

sudo apt-get install python3-virtualenv

Create a virtual environment

python3 -m virtualenv env

Activate a virtual environment

source env/bin/activate

Install pip **#TODO: we have questions about what this is doing pip vs pip3**

python -m pip install --upgrade pip

Install Django

pip3 install django

Create a Helloworld project direction and cd into it

sudo mkdir Helloworld

cd Helloworld

Create a new django project **#TODO We are going to want to be able to test these instructions first, before we jump right into git clone commands.**

django-admin startproject helloworld

cd helloworld

Edit settings.py

sudo vi helloworld/settings.py

Change:

import os

DEBUG = True

ALLOWED\_HOSTS = ['<YOUR\_PUBLIC\_IP>’,’<YOUR\_PUBLIC\_DNS>’]

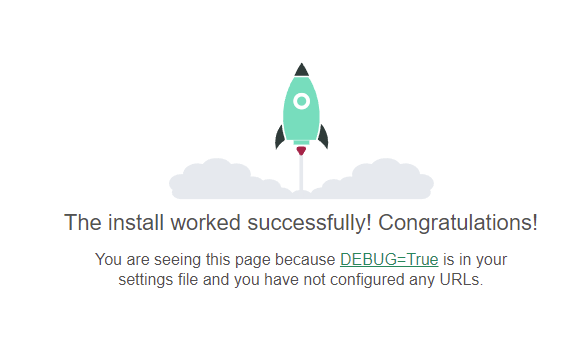
STATIC\_ROOT = os.path.join(BASE\_DIR, "static/")

Test the new Django project and AWS access:

Runserver

python3 manage.py runserver 0.0.0.0:8000

Go to browser URL '<YOUR\_PUBLIC\_IP>:8000'

You should see the Django Rocketship

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

GUNICORN SETUP

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

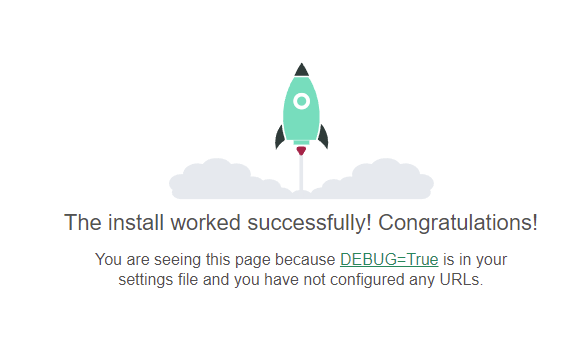
Install the gunicorn wsgi Interface

pip3 install gunicorn

Test it is working

gunicorn --workers 3 --bind 0.0.0.0:8000 helloworld.wsgi:application

Go to browser URL '<YOUR\_PUBLIC\_IP>:8000'

You should see the Django Rocketship

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

NGINX SETUP

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

Install NGINX

sudo apt-get install -y nginx

Start NGINX

sudo nginx

Go to your Browser and paste your AWS domain name

Should now see a message for Welcome to NGINX

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

Integrate Gunicorn with NGINX

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

To integrate Gunicorn with Nginx create a file for NGINX to route incoming requests to the correct project

sudo vi /etc/nginx/sites-available/helloworld

Paste below content in it:

----------------------------------------------------------------

server {

listen 80;

server\_name <YOUR\_PUBLIC\_IP>;

return 302 $scheme://<YOUR\_PUBLIC\_DNS>$request\_uri;

}

server {

listen 80;

# This server\_name should be based on the ending of your <YOUR\_PUBLIC\_DNS>

server\_name \*.us-east-2.compute.amazonaws.com;

location = /favicon.ico { access\_log off; log\_not\_found off; }

location /static/ {

autoindex on;

alias /home/ubuntu/Helloworld/helloworld/helloworld/static/;

#root /home/ubuntu/Helloworld/helloworld;

}

location / {

include proxy\_params;

proxy\_pass http://unix:/home/ubuntu/Helloworld/helloworld/helloworld.sock;

}

}

----------------------------------------------------------------

Now create the soft link of this file in the sites-enabled directory.

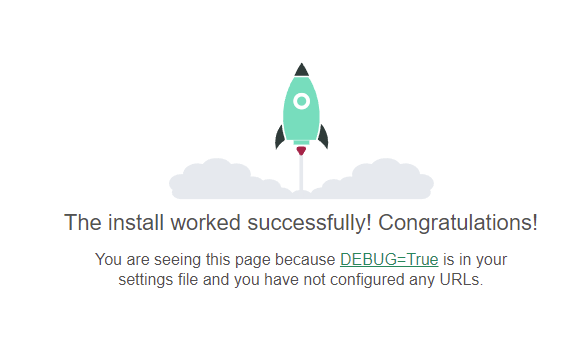
sudo ln -s /etc/nginx/sites-available/helloworld /etc/nginx/sites-enabled

Restart Gunicorn using the below command to enable NGINX and Gunicorn to communicate over a socket instead of a port, which is faster and more secure.

gunicorn --workers 3 --bind unix:/home/ubuntu/Helloworld/helloworld/helloworld.sock helloworld.wsgi:application

Go to browser URL '<YOUR\_PUBLIC\_IP>’ or ‘<YOUR\_PUBLIC\_DNS>’

You should see the Django Rocketship and the PUBLIC IP URL should have redirected to <YOUR\_PUBLIC\_DNS>



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

Adding Supervisor

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

Setup Supervisor to make sure gunicorn is always up and running:

from Terminal under your Project folder (cd /Helloworld/helloworld)

sudo apt-get install -y supervisor

Create supervisor configuration

cd /etc/supervisor/conf.d/

Create gunicorn.conf file

sudo vi gunicorn.conf

Add coding lines

----------------------

[program:gunicorn]

directory=/home/ubuntu/Helloworld/helloworld

command=/home/ubuntu/env/bin/gunicorn --workers 3 --bind unix:/home/ubuntu/Helloworld/helloworld/helloworld.sock helloworld.wsgi:application

autostart=true

autorestart=true

stderr\_logfile=/var/log/gunicorn/gunicorn.err.log

stdout\_logfile=/var/log/gunicorn/gunicorn.out.log

[group:guni]

programs:gunicorn

----------------------

Create Log directory for Gunicorn

sudo mkdir /var/log/gunicorn

Run Supervisor

sudo supervisorctl reread

sudo supervisorctl update

Verify it is Running

sudo supervisorctl status



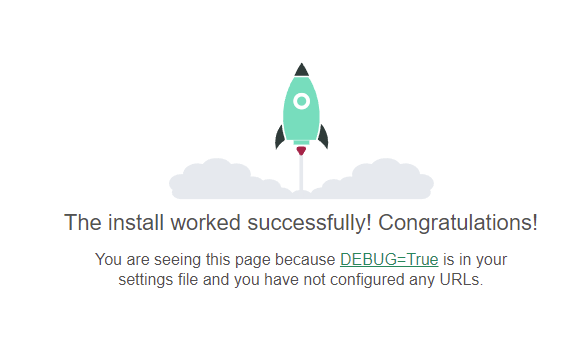
Run the following two commands after making any changes to: The projects code (aka git pull from GitHub), NGINX configuration, Supervisor Configuration,

sudo supervisorctl reload

sudo systemctl reload nginx

Go to browser URL '<YOUR\_PUBLIC\_IP>’ or ‘<YOUR\_PUBLIC\_DNS>’

You should see the Django Rocketship and the PUBLIC IP URL should have redirected to <YOUR\_PUBLIC\_DNS>



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

END - HELLO WORLD SETUP GUIDE COMPLETED - END

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

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

SETUP FOR POSTGRES AND GITHUB PROJECTS

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

Install PostgreSQL:

sudo apt-get update

sudo apt-get install python-dev libpq-dev postgresql postgresql-contrib

Setup PostgreSQL,

sudo su — postgres

Psql

CREATE DATABASE proficere;

Set your password for postgres

/password

Input the password used in your proficere/settings.py file

*# CREATE USER adesh WITH PASSWORD ‘mysecretpass’;*

*# GRANT ALL PRIVILEGES ON DATABASE djangodemodb TO adesh;*

\q

exit

Make sure your Proficere project requirements.txt is updated for the LTS Django, etc. used in AWS (example: Django 3.2.7)

From your virtual environment

source env/bin/activate

To check the AWS version of Django

python3 -m django --version

Clone your Proficere project to your AWS instance

cd

git clone https://github.com/2sduaedanm/Proficere.git

To install from requirements.txt use:

pip install -r requirements.txt

If you don’t have an updated requirements.txt file, install the following

sudo apt-get install libpq-dev

sudo pip3 install Psycopg2

sudo pip install pillow

Run Django migration

cd /Proficere/proficere

python3 manage.py makemigrations

python3 on manage.py migrate

python3 manage.py collectstatic

Load preload data from Proficere

python3 manage.py loaddata accounts/fixtures/dbfixtures.json

Edit Gunicorn and NGINX configuration files

Copy the helloworld config file to proficere

sudo cp /etc/nginx/sites-available/helloworld /etc/nginx/sites-available/proficere

sudo vi /etc/nginx/sites-available/proficere

Replace all forms of helloworld (Helloworld or helloworld) with proficere (Proficere or proficere)

Softlink proficere from sites-available to sites-enabled

sudo ln -s /etc/nginx/sites-available/proficere /etc/nginx/sites-enabled

Edit supervisor

sudo vi /etc/supervisor/conf.d/gunicorn.conf

Replace all forms of helloworld (Helloworld or helloworld) with proficere (Proficere or proficere)

Reload Supervisor and NGINX

sudo supervisorctl reload

sudo systemctl reload nginx

Test with your browser URL '<YOUR\_PUBLIC\_IP>’ or ‘<YOUR\_PUBLIC\_DNS>’ and you should see the Proficere Login screen.

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

Setup SSL/HTTPS and your AWS DNS

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

Update AWS DNS for your instance

Create SSL Certificate

SSH into instance

Ensure that your version of snapd is up to date

sudo snap install core; sudo snap refresh core

Install Certbot

sudo snap install --classic certbot

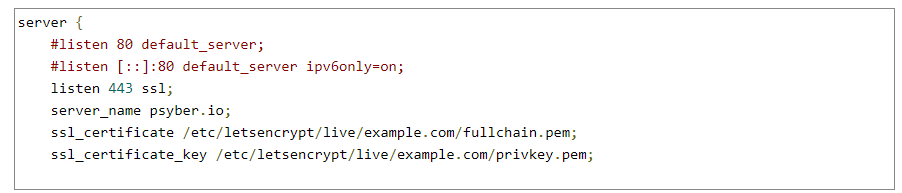
Prepare the certbot command

sudo ln -s /snap/bin/certbot /usr/bin/certbot

Run Certbot

Sudo certbot certonly --nginx

Update Nginx configuration with the certs



Restart the supervisor and Nginx runners

sudo supervisorctl reload

sudo systemctl reload nginx

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

If you AWS Public IP or Host Name Changes:

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

1) Update the IP address and AWS host in Proficere/proficere/settings.py

2) Update the nginx server blocks with new IP address in sudo vi /etc/nginx/sites-available/proficere

3) sudo supervisorctl reload

4) sudo systemctl restart nginx

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

USING LIGHTSAIL:

https://aws.amazon.com/getting-started/hands-on/deploy-python-application/

https://www.youtube.com/watch?v=YyIaseN1KdM

https://www.youtube.com/watch?v=maEvGfJKWdU

https://www.youtube.com/watch?v=sDOlj-j5RZg

https://www.youtube.com/watch?v=mM88-bZgWTc

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

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

OLD STUFF BEYOND HERE

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

Clone Shobi Demo from GitHub

git clone https://github.com/ShobiExplains/AwsDemo.git

Move to the AwsDemo directory

cd AwsDemo

Install the gunicorn wsgi Interface

pip3 install gunicorn

Install NGINX

sudo apt-get install -y nginx

Move to the AwsDemo/helloworld folder and Edit settings.py

cd helloworld

vi settings.py

In settings.py change DEBUG = False

Change ALLOWED\_HOSTS = ['\*'] with the AWS domain name (http://ec2-3-12-155-43.us-east-2.compute.amazonaws.com/) information

This will disable the Local version from working

Start NGINX

sudo ngin

Go to your Browser and paste your AWS domain name

Should now see a message for Welcome to NGINX

Configure gunicorn

Using the AWS "EC2 Connect to Instance":

Reactivate the Virtual Environment

source env/bin/activate

Move to the AwsDemo project folder

cd AwsDemo

Type:

gunicorn --bind 0.0.0.0:8000 YourDjangoFolder.wsgi:application

gunicorn --bind 0.0.0.0:8000 helloworld.wsgi:application

In Browser, your application should now be able to run

Setup Supervisor to make sure gunicorn is always up and running:

from Terminal under your Project folder

sudo apt-get install -y supervisor

Create supervisor configuration

cd /etc/supervisor/conf.d/

sudo touch gunicorn.conf

Edit gunicorn.conf

sudo nano gunicorn.conf

Add coding lines

[program:gunicorn]

directory=/home/ubuntu/YourDjangoFolder

command=/home/ubuntu/env/bin/gunicorn --workers 3 --bind ujnix:/home/ubuntu/YourProjectFolder/app.sock YourDjangoProjectFolder.wsgi:application

autostart=true

autorestart=true

stderr\_logfile=/var/log/gunicorn/gunicorn.err.log

stdout\_logfile=/var/log/gunicorn/gunicorn.out.log

[group:guni]

programs:gunicorn

Ctrl-O will save the file

Ctrl-X will exit

Run supervisor

cd /etc/supervisor/conf.d/

Create directory

sudo mkdir /var/log/gunicorn

Run

sudo supervisorctl reread

sudo supervisorctl update

Check to see if running

sudo supervisorctl status

Configure NGINX

Go to Home directory

cd /etc/nginx/sites-available

ls

see default diretory

Show a sample file

cat default

Create a new file

sudo touch django.conf

Edit Django Configuration file

sudo nano django.conf

Add the following code lines

server {

listen 80;

server\_name YourAWSDomainName.com;

location / {

include proxy\_params;

proxy\_pass http://unix:/home/ubuntu/YourProjectFolder/app.sock

}

}

Save file (ctrl-O and ctrl-X)

Test the Django configuration

Under the /etc/nginx/sites-available

sudo nginx -t

Enable the django.conf

sudo ln django.conf /etc/nginx/sites-enabled/

sudo nginx -t

Restart NGINX

sudo service nginx restart

Your Browser should now show your Site as working

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

HANDLING STATIC FILES: https://www.youtube.com/watch?v=\_TBw7ALJp0Y&list=PLX4uXM5lVU53JbQ\_1ijxpU0qZIOrJOG--&index=2

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

After moving a new version from GitHub (git pull), run

sudo supervisorctl reload

sudo systemctl reload nginx

Open AWS Console

Edit NGINX Configuration

sudo nano /etc/nginx/sites-enabled/django.conf

Add below "location" section

location /static/ {

autoindex on;

alias /home/ubuntu/YourProjectFolder/YourDjangoProject/static/;

}

Save with Ctl-O and Exit with Ctl-X

Reload NGINX

sudo systemctl reload nginx

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

RDS (PostgreSQL): https://www.youtube.com/watch?v=PCjeBQ2636Y&list=PLX4uXM5lVU53JbQ\_1ijxpU0qZIOrJOG--&index=3

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

Got to AWS Console

Choose "RDS"

Click on "create database"

Choose "PostgreSQL"

Check "Only enable options eligible for RDS Free Usage Tier"

Click "Next"

Under Settings:

DB Instance-identifier

"YourDjangoDatabaseIdentifier"

Master username

"YourMasterUserName"

Master password & Confimation

"YourMasterPassword"

Click "Next"

Public accessibility

Public

VPC security groups

Choose existing VPC security groups

default

Database name

"YourDatabaseName:

Click on "Create database"

Once DB is created:

Click on "View DB instance details"

Change settings.py "DATABASES" section

Update

'Name': 'YourProjectDatabaseName'

'USER': 'postgres'

'PASSWORD': 'YourAWSrdsPassword'

'HOST': 'AWSdatabaseEndpoint'

'PORT': 'AWSdatabasePort'

Install psycopg2

sudo pip install psycopg2-binary

Go to your Terminal

Activate your VENV virtual environment

Run python3 manage.py makemigrations YourDjangoProject

Run python3 manage.py migrate YourDjangoProject