Tek Kshetri
Posted on 26 dic 2020 • Updated on 11 feb 2021

Web-GIS Deployment in Ubuntu server (Nginx + Tomcat + PostGIS)

#django #postgis #geoserver #webgis

If you are looking for the web-GIS application deployment technique, read this full blog to get the overall idea about it. The table of content of this blog is as below,

1. Basic server setup

Updating and upgrading the ubuntu server

The `apt update` updates the package lists for upgrades for packages that need upgrading, as well as new packages that have just come to the repositories.

```
sudo apt update
sudo apt upgrade
```

Download SSH

The SSH allows us to connect the server remotely. The ssh can be install by using `sudo apt install openssh-server` and by default, it will start on port 22.

```
sudo apt install openssh-server
```

You can check the status of ssh by following command

```
sudo systemctl status ssh
```

Setup firewall with UFW

UFW, or Uncomplicated Firewall, is an interface to iptables that is geared towards simplifying the process of configuring a firewall. The detailed documentation about UFW can be found here.

The main syntax for allow farewall is `sudo ufw allow from <spacific ip/subnet> to any port <port number>.`

Lets setup the required ports through ufw command;

```
sudo ufw enable
sudo ufw allow ssh
sudo ufw allow http
sudo ufw allow 8080
sudo ufw allow 8000
sudo ufw allow from any to any port 80
```

To check the status of ufw, you can type,

```
sudo ufw status
```

2. Installation of essential tools

In this section, we are going to install the some essential tools like gdal, virtualenv, git, vim, etc. Run the following code to install the required packages,

```
sudo apt update
sudo apt upgrade

# Installation of gdal
sudo apt install -y gdal-bin libgdal-dev libgeos-dev libproj-dev

# Installation of python and virtualenv
```

```
sudo apt install -y python3-pip python3-dev python3-virtualenv python3-venv virtualenv

# Other essential tools
sudo apt install -y software-properties-common build-essential
sudo apt install -y git unzip gcc zlib1g-dev libgeos-dev libproj-dev
sudo apt install -y vim

# Install Openjdk
sudo apt install openjdk-8-jdk-headless default-jdk-headless -y
sudo update-java-alternatives --jre-headless --jre --set java-1.8.0-openjdk-amd64
```

During this installation, python, gdal, java are the essential components. So make sure these packages are installed successfully,

```
# Verify GDAL version
gdalinfo --version
  $> GDAL 3.0.4, released 2020/01/28

# Verify Python version
python3.8 --version
  $> Python 3.8.5

which python3.8
  $> /usr/bin/python3.8

# Verify Java version
java -version
  $> openjdk version "1.8.0_265"
  $> OpenJDK Runtime Environment (build 1.8.0_265-8u265-b01-0ubuntu2~20.04-b01)
  $> OpenJDK 64-Bit Server VM (build 25.265-b01, mixed mode)
```

Virtualenv Setup

Now let's create the python virtualenv and setup it properly. I am going to use virtualwrapper for this purpose.

```
# Create the Virtual Environment named as 'venv' (first time only)
source /usr/share/virtualenvwrapper/virtualenvwrapper.sh
mkvirtualenv --python=/usr/bin/python3 venv

# Activate virtual env
source /usr/share/virtualenvwrapper/virtualenvwrapper.sh
workon venv
```

In order to save it permanently, edit the `.bashrc` file and following line of code at the bottom of the file,

```
nano ~/.bashrc

# Write to the bottom of the file the following lines
export WORKON_HOME=~/.virtualenvs
source /usr/share/virtualenvwrapper/virtualenvwrapper.sh
```

3. PostGIS database setup

In this section, we are going to install the PostgreSQL with PostGIS extension for data storage.

```
# Ubuntu 20.04 (focal)
sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt/ `lsb_release -cs`-pgdg
sudo wget --no-check-certificate --quiet -O - https://www.postgresql.org/media/keys/AC
sudo apt update -y; sudo apt install -y postgresql-13 postgresql-13-postgis-3 postgres
```

In the above code, the first and second line of code will add the PostgreSQL to the apt repository. After that, the third line of code will install the PostgreSQL 13 with PostGIS 3 version.

After successful installation of PostgreSQL, you can start the PostgreSQL server by following command,

```
sudo service postgresql start
```

For checking the status of PostgreSQL server, you can run following command,

```
sudo service postgresql status
```

Now let's create the database named as `geoapp` and create the PostGIS extension on it by running following command.

```
sudo -u postgres createdb -O geoapp
sudo -u postgres psql -d geoapp -c 'CREATE EXTENSION postgis;'
```

If you have the database backup file, you can restore it into the server database as well. I suppose you have a database backup file named as `backup.sql`. (**This step is not necessary if you don't care about the development data**)

```
psql -U postgres -h localhost -p 5432 -d geoapp -f backup.sql
```

4. GeoServer installation

In computing, GeoServer is an open-source server written in Java that allows users to share, process and edit geospatial data. In this section, we are going to install the GeoServer in tomcat server.

Tomcat installation

Tomcat is an open-source implementation of the java servlet. Since we already installed the java in previous section, In this section we are going to install the tomcat server on open jdk.
First of all let's create the dedicated user `tomcat` and add it to the group `www-data`,

```
sudo useradd -m -U -d /opt/tomcat -s /bin/bash tomcat
sudo usermod -a -G www-data tomcat
```

The first line of code creates the user named as `tomcat`.
-m, --create-home: Create the user's home directory if it does not exist.
-U, --user-group: Create a group with the same name as the user, and add the user to this group.
-d, --home HOME_DIR: The new user will be created using HOME_DIR as the value for the user's login directory (i.e. /opt/tomcat).
-s, --shell: The name of the user's login shell.
-a, --append: Add the user to the supplementary group(s). Use only with the -G option.
-G, --groups: A list of supplementary groups which the user is also a member of. Each group is separated from the next by a comma, with no intervening whitespace.

Now let's download the tomcat server. In this tutorial, I am using `9.0.41 version` tomcat. The downloaded tomcat file has to unzip inside `/opt/tomcat` dir.

```
VERSION=9.0.41; wget https://www-eu.apache.org/dist/tomcat/tomcat-9/v${VERSION}/bin/ap
sudo tar -xf apache-tomcat-${VERSION}.tar.gz -C /opt/tomcat/; rm apache-tomcat-${VERSI
```

Since the tomcat is updated regularly. So, to have more control over version and updates, we'll create the symbolic link as below:

```
sudo ln -s /opt/tomcat/apache-tomcat-${VERSION} /opt/tomcat/latest
```

Now let's change the owner and mode of /opt/tomcat directory.

```
sudo chown -R tomcat:www-data /opt/tomcat/
sudo sh -c 'chmod +x /opt/tomcat/latest/bin/*.sh'
```

Create the symbolic link for JRE (java) as well,

```
sudo ln -s /usr/lib/jvm/java-8-openjdk-amd64/jre/ /usr/lib/jvm/jre
```

We want to be able to run Tomcat as a service, so we will set up systemd service file. Tomcat need to know where Java is installed.

Now we can create the system service file, which can control over the tomcat services (start, stop, restart, enable).

```
sudo vim /usr/lib/systemd/system/tomcat9.service
```

And write the following line of code to the file,

```
[Unit]
Description=Apache Tomcat Server
After=syslog.target network.target

[Service]
Type=forking
User=tomcat
Group=tomcat

Environment=JAVA_HOME=/usr/lib/jvm/jre
Environment=JAVA_OPTS=-Djava.security.egd=file:///dev/urandom
Environment=CATALINA_PID=/opt/tomcat/latest/temp/tomcat.pid
Environment=CATALINA_HOME=/opt/tomcat/latest
Environment=CATALINA_BASE=/opt/tomcat/latest

ExecStart=/opt/tomcat/latest/bin/startup.sh
ExecStop=/opt/tomcat/latest/bin/shutdown.sh

RestartSec=30
Restart=always

[Install]
WantedBy=multi-user.target
```

When you are finished, save and close the file.

Next, reload the systemd daemon so that it knows about our service file:

```
sudo systemctl daemon-reload
```

Enable the tomcat server and start it,

```
sudo systemctl enable tomcat9.service
sudo systemctl start tomcat9.service
```

To check status of the tomcat service,

```
sudo systemctl status tomcat9.service
```

GeoServer Installation

Download the stable version of geoserver.war file and paste it into webapps dir of the tomcat. Here, we are going to download the geoserver.war file from geosolution.it.

```
cd /tmp
sudo wget --no-check-certificate https://build.geo-solutions.it/geonode/geoserver/late
sudo mv geoserver-2.17.2.war /opt/tomcat/latest/webapps/geoserver.war
```

Now the geoserver will run in the port 8080 of you DNS (http://localhost:8080/geoserver)

This is the general installation guide of geoserver in ubuntu server. If you want to have a separated data dir for geoserver, check this guide.

5. Django project implementation

Here I am going to deploy the simple GeoDjango project from GitHub repo.

```
sudo mkdir /opt/geoProject

# Change owner and user of this folder
sudo chown -Rf $USER:www-data /opt/geoProject/
sudo chmod -Rf 775 /opt/geoProject/

cd /opt/geoProject
git clone https://github.com/iamtekson/geodjango-app.git /opt/geoProject
```

Before starting the installation of dependencies, we can activate the virtualenv,

```
workon venv
```

Now, we can install the dependencies.

```
pip install -r requirements.txt
pip install gunicorn
pip install pygdal=="`gdal-config --version`.*"
```

Update on settings.py file

Since our setting on development and production mode may be different, we need to update the settings.py file,

```
nano /opt/geoPorject/settings.py
```

Make changes on the ALLOWED_HOSTS field,

```
# The simplest case: just add the domain name(s) and IP addresses of your Django serve
# ALLOWED_HOSTS = [ 'example.com', '203.0.113.5']
# To respond to 'example.com' and any subdomains, start the domain with a dot
# ALLOWED_HOSTS = ['.example.com', '203.0.113.5']
ALLOWED_HOSTS = ['your_server_domain_or_IP', 'second_domain_or_IP', . . .]
```

Changes on database connection parameters,

```
DATABASES = {
    'default': {
        'ENGINE': 'django.contrib.gis.db.backends.postgis',
        'NAME': 'geoapp',
        'USER': 'postgres',
        'PASSWORD': 'admin',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}
```

Now, we can migrate the database and createsuperuser by typing following code,

```
python manage.py makemigrations
python manage.py migrate
python manage.py collectstatic
python manage.py createsuperuser
```

Now you can test the Django server,

```
python manage.py runserver 0.0.0.0:8000
```

6. Nginx and gunicorn configuration

Gunicorn Configuration

Gunicorn 'Green Unicorn' is a Python WSGI HTTP Server for UNIX. It's a pre-fork worker model. The Gunicorn server is broadly compatible with various web frameworks, simply implemented, light on server resources, and fairly speedy. Now we can test the gunicorn's ability to serve the project by following command,

```
cd /opt/geoProject
gunicorn --bind 0.0.0.0:8000 geoProject.wsgi
```

This will start Gunicorn on the same interface that the Django development server was running on. You can go back and test the app again.

Now we can deactivate the virtualenv by typing:

```
deactivate
```

Create a Gunicorn systemd Service File

We have tested the Gunicorn can interact with our Django application, but we should implement a more robust way of starting and stopping the application server. To accomplish this, we'll make a systemd service file.

Create and open the systemd file for gunicorn,

```
sudo nano /etc/systemd/system/gunicorn.service
```

For this demo, my django project name is `geoProject` and my virtualenv name is `venv` as we created in first section.
Add the following line of code inside `gunicorn.service` file.

```
[Unit]
Description=gunicorn daemon
After=network.target

[Service]
User=$USER
Group=www-data
```

```
WorkingDirectory=/opt/geoProject
ExecStart=/home/$USER/.virtualenvs/venv/bin/gunicorn --access-logfile - --workers 3 --

[Install]
WantedBy=multi-user.target
```

We can now start the Gunicorn service we created and enable it so that it starts at boot:

```
sudo systemctl daemon-reload
sudo systemctl start gunicorn
sudo systemctl enable gunicorn
```

Nginx setup

Nginx is a web server that can also be used as a reverse proxy, load balancer, mail proxy and HTTP cache. The Nginx can be installed using the following command,

```
sudo apt install -y nginx
```

Now lets create and open the file `myproject` inside `/etc/nginx/sites-available/geoproject`,

```
sudo nano /etc/nginx/sites-available/geoproject
```

Make sure, you change the `server_domain_or_IP` to your actual domain or IP, otherwise, it will only work on localhost.

```
upstream geoserver_proxy {
  server localhost:8080;
}

server {
    listen 80;
    server_name server_domain_or_IP;

    location = /favicon.ico { access_log off; log_not_found off; }
    location /static/ {
        root /opt/geoProject;
    }

    location / {
        include proxy_params;
        proxy_pass http://unix:/opt/geoProject/geoProject/geoProject.sock;
```

```
    }
    location /geoserver {
        proxy_pass http://geoserver_proxy;
        include proxy_params;
    }
}
```

Save and close the file when you are finished.
We started by specifying that this block should listen on the normal port 80 and that it should respond to our server's domain name or IP address.
Next, we tell Nginx to ignore any problems with finding a favicon. After that, we collected the static assets in `/opt/geoPorject/static` directory.
Next, we added the geoserver proxy so that we can access the geoserver through this URL: http://localhost/geoserver (without assigning the 8080 port in URL).
Finally, we created a location / {} block to match all other requests. Inside of this location, we'll include the standard proxy_params file included with the Nginx installation and then we will pass the traffic to the socket that our Gunicorn process created.

Now we can enable the file by linking it to the sites-enabled directory (i.e. create a symbolic link to the site-available directory).

```
sudo ln -s /etc/nginx/sites-available/geoproject /etc/nginx/sites-enabled
```

Delete the default Nginx file from `site-enabled` and `site-available` directory,

```
sudo rm -rf /etc/nginx/sites-available/default
sudo rm -rf /etc/nginx/sites-enabled/default
```

Test the Nginx configuration for syntax error by typing;

```
sudo nginx -t
```

If no error happens, restart the server by typing;

```
sudo systemctl restart nginx
```

In case of error happen, check the nginx log file,

```
sudo tail -F -n 300 /var/log/nginx/error.log
```

Finally, we need to open up our firewall to normal traffic on port 80. Since we no longer need 8000 port, we can remove it;

```
sudo ufw delete allow 8000
sudo ufw allow 'Nginx Full'
```

Now you will be able to view your application on your server's domain or IP address. If your domain name is example.com, then your application will run on the following URL:

http://example.com

and the GeoServer will run on the following URL:
http://example.com/geoserver

Thank you for reading my blog. If you want to learn more about the web-Mapping and web-GIS things, please subscribe my youtube channel.

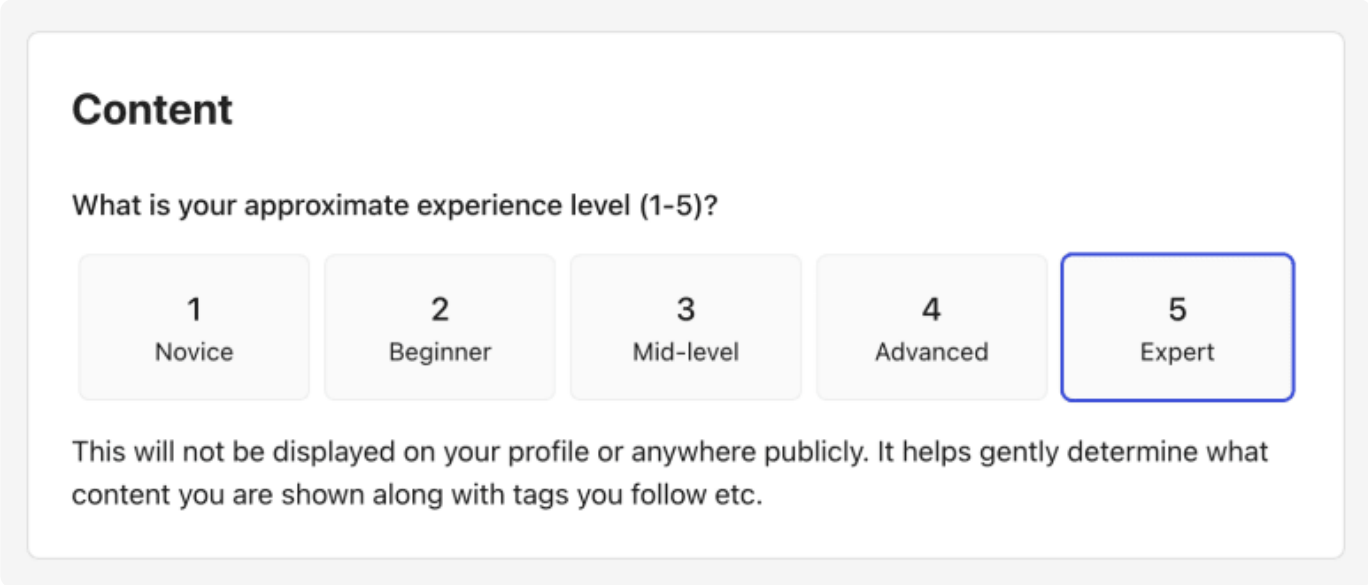
Top comments (1)

ferchimi • 9 sept 21

Hi, thanks for the tutorial
I could't finish the right installation, at this part:
python manage.py makemigrations
python manage.py migrate
python manage.py collectstatic <---- fails
python manage.py createsuperuser

FileNotFoundError: [Errno 2] No such file or directory: '/opt/geoProject/geoProject/static'
I could not find the static content anywhere, please help me find what have gone wrong
Regards
Fernando

Take a look at this:



Go to your customization settings to nudge your home feed to show content more relevant to your developer experience level. 🛠️

Tek Kshetri

I am a Geomatics engineer working as a research associate in Geoinformatics Center (GIC), AIT, Thailand. I am mainly passionated for spatial data analysis and web-GIS development.

LOCATION
Thailand

WORK
Research Associate at Asian Institute of Technology (AIT)

JOINED
10 dic 2019

More from Tek Kshetri

How to migrate database into production server in Django
#django #python #migration #postgres

WFS request in Geoserver using Leafletjs
#geoserver #javascript #leafletjs #wfs

Get extent of WMS layer dynamically in Leafletjs
#javascript #leaflet #geoserver #wms