How to Point Domain and Host Django Project on Nginx Web Server

- Get Access to Remote Server via SSH

```
Syntax:- ssh -p PORT USERNAME@HOSTIP
Example:- ssh -p 22 raj@216.32.44.12
```

Note:- Run Below Commands on Your Remote Server Linux Machine or VPS, Not on Your Local Windows Machine

- Verify that all required softwares are installed

```
nginx -v
  python --version
  pip --version
- SQLite is Included with Python
  python -c "import sqlite3; print(sqlite3.sqlite_version)"
```

- If Required Softwares and Modules are not Installed then Install them:

```
sudo apt install nginx
sudo apt install python
sudo apt install libapache2-mod-wsgi-py3
sudo apt install python3-pip
```

- Install virtualenv

```
pip list
sudo pip install virtualenv
```

- Verify Apache2 is Active and Running

```
sudo systemctl nginx restart
```

- Verify Web Server Ports are Open and Allowed through Firewall

```
sudo ufw status verbose
```

- Go to Your Project Directory

```
Syntax:- cd /var/www/project_folder_name
Example:- cd /var/www/hires_api
```

- Create Virtual env

```
Syntax:- virtualenv env_name
Example:- virtualenv env
```

- Activate Virtual env

```
Syntax:- source virtualenv_name/bin/activate
Example:- source env/bin/activate
```

- Install Dependencies

```
pip install -r requirements.txt
```

How to Install PostgreSQL On Ubuntu

Follow the steps in the sections below to install PostgreSQL from the PostgreSQL repository.

Step 1: Add PostgreSQL Repository

Run the following command to add the PostgreSQL repository to your system:

```
sudo sh -c 'echo "deb https://apt.postgresql.org/pub/repos/apt
$(lsb_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'
```

Step 2: Add the Repository Signing Key

The next step is to fetch the repository's GPG key and add it to APT's trusted keyring. This allows APT to verify the authenticity of packages downloaded from the PostgreSQL repository.

Run the command below:

```
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc |
sudo apt-key add -
```

Step 3: Update the Package List

After adding the official PostgreSQL repository, update the package list to ensure you install the latest PostgreSQL package.

```
sudo apt update
```

Step 4: Install PostgreSQL

To install **PostgreSQL** run the following command:

```
sudo apt install postgresql
```

Verify PostgreSQL Installation

Verify that PostgreSQL has been installed by checking the PostgreSQL service status. Run the command below:

```
sudo systemctl status postgresql
```

Connect to PostgreSQL

To establish a connection with the database, log into the **`postgres`** account with:

```
sudo -i -u postgres
```

Next, open a **`postgres`** prompt using the command:

```
psql
```

Use the following syntax to create a postgres password:

```
\password postgres
```

You will be prompted to type a new password. Repeat this for the owner and postgres user, giving each a strong, unique password.

To exit psql, type

```
\q.
```

Use the following syntax to create a database:

```
Syntax: CREATE DATABASE [dbname];
Example: CREATE DATABASE hires_database;
```

Check Connection Information

If you are connected to PostgreSQL and want to see details of the connection, use the command:

```
\conninfo
```

Use the following syntax to create a user and password:

```
Syntax: CREATE USER <username> with ENCRYPTED PASSWORD '<password>';
Example: CREATE USER hires with ENCRYPTED PASSWORD 'postgres@123';
```

Use the following syntax to create a user and password:

```
Syntax: ALTER DATABASE <dbname> OWNER TO <username>;
ALTER DATABASE hires_database OWNER TO hires;
```

To exit psql, type

```
\q.
```

Allow PostgreSQL TCP port 5432 in the Firewall

PostgreSQL default HTTP port is 5432, you'll need to allow access to this port on the firewall.

If your firewall is UFW type the following commands:

```
sudo ufw allow 5432/tcp
```

Add PostgreSQL Server to pgAdmin

Check PostgreSQL Configuration

1. Verify postgresql.conf:
   Locate the postgresql.conf file. Its location can vary, but it's often in

```
cd /etc/postgresql/16/main/
sudo nano postgresql.conf
```

```
          listen_addresses = '*'
```

2. Verify pg_hba.conf:
   Locate the pg_hba.conf file. Its location can vary, but it's often in

```
cd /etc/postgresql/16/main/
sudo nano pg_hba.conf
```

```
#IPv4 Addresses
host all all 0.0.0.0/0 md5
#IPv6 Addresses
host all all ::0/0 md5
```

3. Restart PostgreSQL configuration to apply changes:

```
sudo service restart postgresql
```

4. Reload PostgreSQL configuration to apply changes:

```
sudo service reload postgresql
```

Add PostgreSQL Server to pgAdmin

In pgAdmin, click on the “Servers” menu on the left sidebar, then right-click and choose “Register” then select “Server” Enter a name for your server and switch to the “Connection” tab. Fill in the following details:

Host name/address: <server-ip address>

Port: 5432

Maintenance database: postgres

Username: postgres

Password: (Enter the password you set for the PostgreSQL user)

Click “Save” to add the server.

Congratulations! You’ve successfully installed and configured PostgreSQL and pgAdmin on your Ubuntu system. You can now use pgAdmin to manage your databases, tables, and perform various administrative tasks through its intuitive interface.

- Serve Static Files

```
cd /var/www/hires_backend
python manage.py collectstatic
```

- Create Database Tables

```
python manage.py makemigrations
python manage.py migrate
```

- Create Superuser

```
python manage.py createsuperuser
```

- If Database File throws error Permission Denied then Set below permissions
- Make Webserver as owner for database file. Our Webserver is running as www-data and group is also www-data.

```
Syntax:-
sudo chown -R www-data:www-data database_folder
sudo chmod 775 database_folder
sudo chmod 664 database_folder/database_file

Example:-
sudo chown -R www-data:www-data mbdb
sudo chmod 775 mbdb
sudo chmod 664 mbdb/db.sqlite3
```

- If Media Files (User Uploaded Files) throws error Permission Denied then Set below permissions

```
sudo chown -R www-data:www-data media
```

- If needed Deactivate Virtual env

```
deactivate
```

- Create hires.ini file

    1. **The file projectname.ini is likely a configuration file for uWSGI. The projectname.ini file extension suggests it's an INI-style configuration file used to define how uWSGI should run and serve your application.**

    2. **In your case, the path suggests it might be set up to handle the uWSGI configuration for a site called projectname. The sites directory in /etc/uwsgi/ is commonly used for configuration files for different applications or sites.**

- Create a file of hires.ini

```
Syntax:- sudo nano /etc/uwsgi/sites/projectname.ini
Example:- sudo nano /etc/uwsgi/sites/hires.ini
```

- Write a below code on hires.ini

```
[uwsgi]
project = hires
uid = www-data
gid = www-data
base = /var/www/hires-api

chdir = %(base)/hires
home = %(base)/env
module = wsgi:application

master = true
processes = 5

socket = /run/uwsgi/%(project).sock
chown-socket = www-data:www-data
chmod-socket = 660
vacuum = true

die-on-term = true
```

- Create uwsgi.service file

  The uwsgi.service is a systemd service unit file that is used to manage uWSGI as a service on a Linux system. uWSGI is a popular application server for deploying Python web applications. It can communicate with various web servers using protocols like WSGI, FastCGI, and others. When you install uWSGI, you often create a uwsgi.service file to configure how uWSGI should start, stop, and restart with the system.

```
Syntax:- sudo nano /etc/systemd/system/uwsgi.service
Example:- sudo nano /etc/systemd/system/uwsgi.service
```

- Write a below code on uwsgi.service

```
[Unit]
Description=uWSGI Emperor service

[Service]
ExecStart=/usr/bin/uwsgi --ini /etc/uwsgi/sites/hires.ini
Restart=always
KillSignal=SIGQUIT
Type=notify
NotifyAccess=all

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

- Relaod deamon

```
sudo systemctl daemon-reload
```

- Start uwsgi

```
sudo systemctl start uwsgi
```

- Enable uwsgi

```
sudo systemctl enable uwsgi
```

- Restart nginx

```
sudo systemctl restart nginx
```

- Create Virtual Host File

```
Syntax:- sudo nano /etc/nginx/conf.d/your_domain.conf
Example:- sudo nano /etc/nginx/conf.d/hires.broaderai.com.conf
```

```
# Define the upstream server
upstream hires_app {
    server 127.0.0.1:8000;
}

# HTTPS server block
server {
    listen 443 ssl; # managed by Certbot
    server_name hires.broaderai.com;

#    add_header Access-Control-Allow-Origin "*";


    # SSL configuration
    ssl_certificate /etc/letsencrypt/live/hires.broaderai.com/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/hires.broaderai.com/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

    # Static files
    location /static/ {
        alias /var/www/hires-api/static/;
        add_header Access-Control-Allow-Origin "*";
    }

    # Media files
    location /media/ {
        alias /var/www/hires-api/media/;
        add_header Access-Control-Allow-Origin "*";
        add_header Access-Control-Allow-Methods "GET, OPTIONS";
        add_header Access-Control-Allow-Headers "Authorization, Content-Type";
    }
```

```
    # Media files
    location /media/ {
        alias /var/www/hires-api/media/;
        add_header Access-Control-Allow-Origin "*";
        add_header Access-Control-Allow-Methods "GET, OPTIONS";
        add_header Access-Control-Allow-Headers "Authorization, Content-Type";
    }

    # Proxy pass to the application server
    location / {
        proxy_pass http://hires_app;
        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;
    }

    # Logging
    error_log /var/log/nginx/hires_error.log;
    access_log /var/log/nginx/hires_access.log;
}

# HTTP server block (redirect to HTTPS)
server {
    if ($host = hires.broaderai.com) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


    listen 80;
    server_name hires.broaderai.com;

    # Redirect all HTTP requests to HTTPS
    if ($scheme != "https") {
        return 301 https://$server_name$request_uri;
    }

    # Fallback for non-HTTPS requests (should not be used if Certbot is managing SSL)
    return 404; # managed by Certbot
}
```

- Check Configuration is correct or not

```
sudo nginx -t
```

- Enable Virtual Host

```
cd /etc/nginx/conf.d
sudo a2ensite hires.broaderai.com.conf
```

- Restart nginx

```
sudo systemctl restart nginx
```

- Reload nginx

```
sudo systemctl reload nginx
```

Supervisor configuration on linux

- Install Supervisor

```
sudo apt-get install supervisor
```

- To Configure our supervisor, we will have to create few configuration files and set them up

  Example:

```
# open editer
sudo nano /etc/supervisor/conf.d/myproject.conf

# copy paste the command and modify your project location
[program:myproject]
command=/path/to/your/venv/bin/gunicorn myproject.wsgi:application -
workers 3 - bind 0.0.0.0:8000
directory=/path/to/your/django/project
autostart=true
autorestart=true
stderr_logfile=/var/log/myproject.err.log
stdout_logfile=/var/log/myproject.out.log

# directory: Specifies the working directory of your Django project.
# autostart and autorestart: Ensure that Supervisor starts and restarts
the process automatically.
# stderr_logfile and stdout_logfile: Log files for standard error and
standard output.
```

- Create Supervisor configuration on linux

```
Syntax:- sudo nano /etc/supervisor/conf.d/myproject.conf
```

```
Example:- sudo nano /etc/supervisor/conf.d/hires.conf
```

```
#example of your django project
sudo nano /etc/supervisor/conf.d/hires.conf

[program:hires]
command=/var/www/hires-api/env/bin/gunicorn hires.wsgi:application -
workers 3 - bind 0.0.0.0:8000
directory=/var/www/hires-api
autostart=true
autorestart=true
stderr_logfile=/var/log/hires.err.log
stdout_logfile=/var/log/hires.out.log
```

- Reread the supervisor

```
sudo supervisorctl reread
```

- Update the supervisor

```
sudo supervisorctl update
```

- Status about supervisor

```
sudo supervisorctl status
```

- If you have any change on Django file on your python project then you restart the supervisor via gunicorn

```
Run the following command to restart the specific process:
Syntax: sudo supervisorctl restart app_name
Example: sudo supetvisorctl restart hires
```

- Reread the supervisor

```
sudo supervisorctl reread
```

- Update the supervisor

```
sudo supervisorctl update
```

- Status about supervisor

```
sudo supervisorctl status
```

How to Point Domain and Host ReactJS Project using Nginx Web Server

- To Access Remote Server via SSH

```
Syntax:- ssh -p PORT USERNAME@HOSTIP
Example:- ssh -p 22 raj@216.32.44.12
```

Note:- Run Below Commands on Your Remote Server Linux Machine or VPS, Not on Your Local Windows Machine

- Verify that all required software are installed

```
nginx -v
node -v
npm -v
git --version
```

- Install Apache

```
sudo apt install nginx
```

- Install Node and npm

```
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
&&\
sudo apt-get install -y nodejs
```

- Verify Nginx is Active and Running

```
sudo systemctl status nginx
```

- Go to Your Project Directory

```
Syntax:- cd /var/www/project_folder_name
Example:- cd /var/www/hires-recruiters
```

- Install Dependencies

```
npm install
```

- Create Production Build

```
npm run build
// OR
npm run export
```

- Create Virtual Host File

```
Suntax: sudo nano /etc/nginx/conf.d/your_domain.conf
Example: sudo nano /etc/nginx/conf.d/hires.ohminsights.ca.conf
```

```
server {
    server_name hires.ohminsights.ca;

    # Root directory for the React app
    root /var/www/hires-recruiters/dist;

    # Serve index.html for all requests (for React Router)
    location / {
        try_files $uri /index.html;
    }

    # Logging
    error_log /var/log/nginx/hires_frontend_error.log;
    access_log /var/log/nginx/hires_frontend_access.log;

    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/hires.ohminsights.ca/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/hires.ohminsights.ca/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot


}
server {
    if ($host = hires.ohminsights.ca) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


    listen 80;
    server_name hires.ohminsights.ca;
    return 404; # managed by Certbot


}
```

- Enable Virtual Host

```
cd /etc/nginx/conf.d
sudo a2ensite hires.ohminsights.ca.conf
```

- Check Configuration is correct or not

```
sudo nginx -t
```

- Restart and Reload Nginx

```
sudo systemctl nginx restart

sudo systemctl nginx reload
```

How to Enable HTTPS in Your Domain Hosted on Linux Remote Server or VPS

Let's Encrypt is a non-profit certificate authority run by Internet Security Research Group that provides X.509 certificates for Transport Layer Security encryption at no charge.

- To Access Remote Server via SSH

```
Syntax:- ssh -p PORT USERNAME@HOSTIP
Example:- ssh -p 22 root@216.32.44.12
```

- Install Certbot and python3-certbot-apache

```
sudo apt install certbot python3-certbot-nginx
```

```
- Certbot is a free, open source software tool for automatically using
Let’s Encrypt certificates on manually-administrated websites to enable
HTTPS.
- python3-certbot-nginx is a nginx plugin for Certbot.
```

- Verify Web Server Ports are Open and Allowed through Firewall

```
sudo ufw status verbose
```

- Obtain an SSL certificate

```
sudo certbot --nginx
```

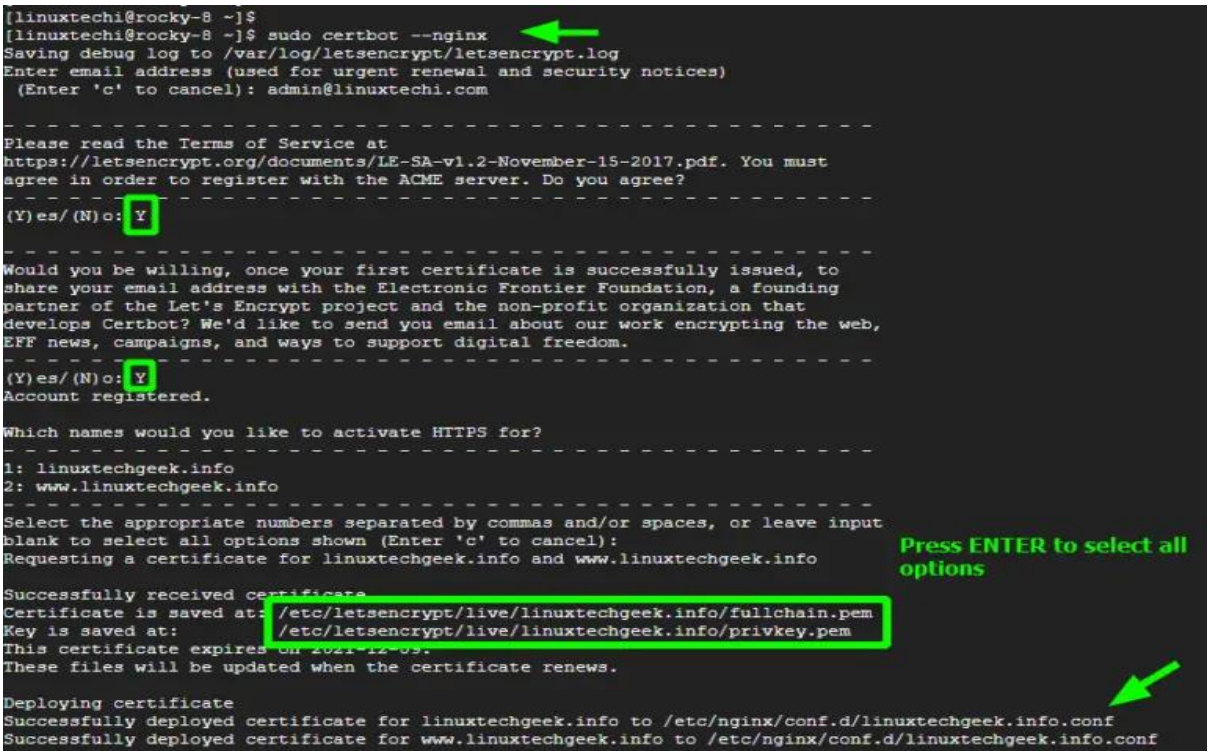
- Check Status of Certbot

```
sudo systemctl status certbot.timer
```

- Dry Run SSL Renewal

```
sudo certbot renew --dry-run
```

Example:-


[linuxtechi@rocky-8 ~]$
[linuxtechi@rocky-8 ~]$ sudo certbot --nginx
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Enter email address (used for urgent renewal and security notices)
(Enter 'c' to cancel): admin@linuxtechi.com
Please read the Terms of Service at
https://letsencrypt.org/documents/LE-SA-v1.2-November-15-2017.pdf. You must
agree in order to register with the ACME server. Do you agree?
(Y)es/(N)o: Y
Would you be willing, once your first certificate is successfully issued, to
share your email address with the Electronic Frontier Foundation, a founding
partner of the Let's Encrypt project and the non-profit organization that
develops Certbot? We'd like to send you email about our work encrypting the web,
EFF news, campaigns, and ways to support digital freedom.
(Y)es/(N)o: Y
Account registered.
Which names would you like to activate HTTPS for?
1: linuxtechgeek.info
2: www.linuxtechgeek.info
Select the appropriate numbers separated by commas and/or spaces, or leave input
blank to select all options shown (Enter 'c' to cancel):
Requesting a certificate for linuxtechgeek.info and www.linuxtechgeek.info
Press ENTER to select all options
Successfully received certificate.
Certificate is saved at: /etc/letsencrypt/live/linuxtechgeek.info/fullchain.pem
Key is saved at: /etc/letsencrypt/live/linuxtechgeek.info/privkey.pem
These files will be updated when the certificate renews.
Deploying certificate
Successfully deployed certificate for linuxtechgeek.info to /etc/nginx/conf.d/linuxtechgeek.info.conf
Successfully deployed certificate for www.linuxtechgeek.info to /etc/nginx/conf.d/linuxtechgeek.info.conf