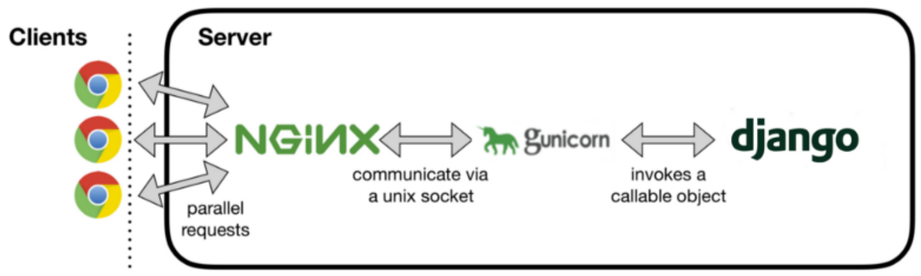
To host Django Application using gunicorn & nginx in Production.

1. **Install Python and Nginx:**



```
sudo apt update
sudo apt install python3-pip python3-dev nginx
```

- python3-pip: This is the Python package installer for Python 3. The **pip tool is used to install Python packages from the Python Package Index (PyPI).** python3-pip is the package that provides the pip tool for Python 3.
- python3-dev: This package contains **files needed for building Python extensions. It includes header files and a static library for Python, which are necessary when building or installing certain Python packages that require compilation.**
- nginx: This is a high-performance web server that is widely used for serving static content, reverse proxying, load balancing, and more. In this context, nginx is one of the packages being installed using the apt package manager.

2. **Creating a Python Virtual Environment:**

```
sudo pip3 install virtualenv
```

- This will install a **virtual environment package in python.** Let's create a project directory to host our Django application and create a virtual environment inside that directory.
- **virtualenv env creates the virtual environment from the virtualenv package we have installed.**
- It helps manage dependencies, improves reproducibility, and ensures a clean and isolated environment for each project.

```
pwd
cd /home
mkdir assginmentdir
cd assignmentdir
virtualenv env
source env/bin/activate
```

3. **Installing Django and gunicorn:**

- This installs Django and gunicorn in our virtual environment.
- **Django is a high-level Python web framework that encourages rapid development and clean, programatic design. Built by experienced developers, it takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.**

```
pip install django gunicorn
```

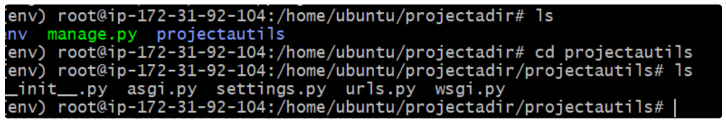
- **Gunicorn is a WSGI compliant web server for Python Applications that receive requests sent to the Web Server from a Client and forwards them onto the Python applications or Web Frameworks (such as Flask or Django) in order to run the appropriate application code for the request.**
- Gunicorn, which stands for "Green Unicorn," is a popular, production-ready WSGI (Web Server Gateway Interface) server for running Python web applications.

- It is designed to work with various web frameworks, and one of its common use cases is to serve Django applications. Gunicorn is a pre-fork worker model server, which means it can handle multiple requests concurrently by creating multiple worker processes.
- Nginx handles tasks such as serving static files, managing SSL, and acting as a reverse proxy, Gunicorn focuses on executing Python code and handling WSGI communication.
- The combination of Nginx and Gunicorn is a well-established and widely used architecture for deploying scalable and performant Python web applications in production environments.

4. **Setting up our Django project:**

- This command **creates a project named assignutils in /home/assignmentdir, and creates manage.py file.**
- django-admin command is used as command line utility for managing django projects. It provides various common tasks such as **creating projects, runing development servers and creating db tables.**
- startproject is the sub command to create the project.

```
django-admin startproject assignutils /home/assignmentdir
```



```
(env) root@ip-172-31-92-104:/home/ubuntu/projectadir# ls
env  manage.py  projectautils
(env) root@ip-172-31-92-104:/home/ubuntu/projectadir# cd projectautils
(env) root@ip-172-31-92-104:/home/ubuntu/projectadir/projectautils# ls
_init__.py  asgi.py  settings.py  urls.py  wsgi.py
(env) root@ip-172-31-92-104:/home/ubuntu/projectadir/projectautils# |
```

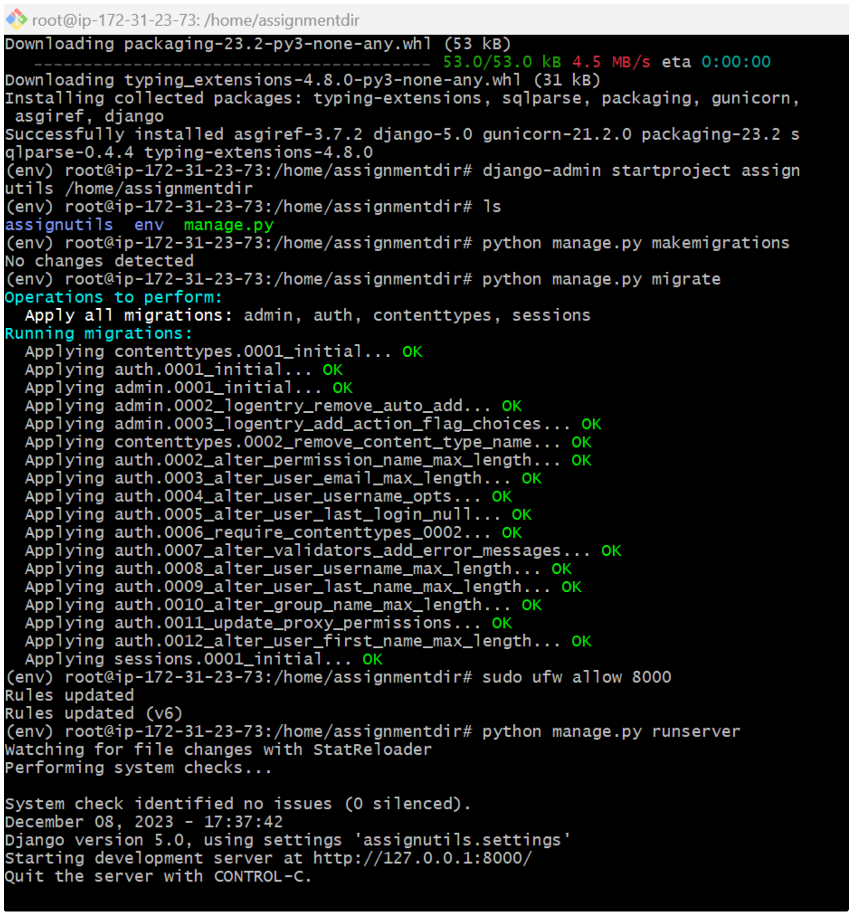
Add your IP address or domain to the ALLOWED_HOSTS variable in settings.py.

```
python manage.py makemigrations
python manage.py migrate
sudo ufw allow 8000
python manage.py runserver
python manage.py runserver 0.0.0.0:8000
```

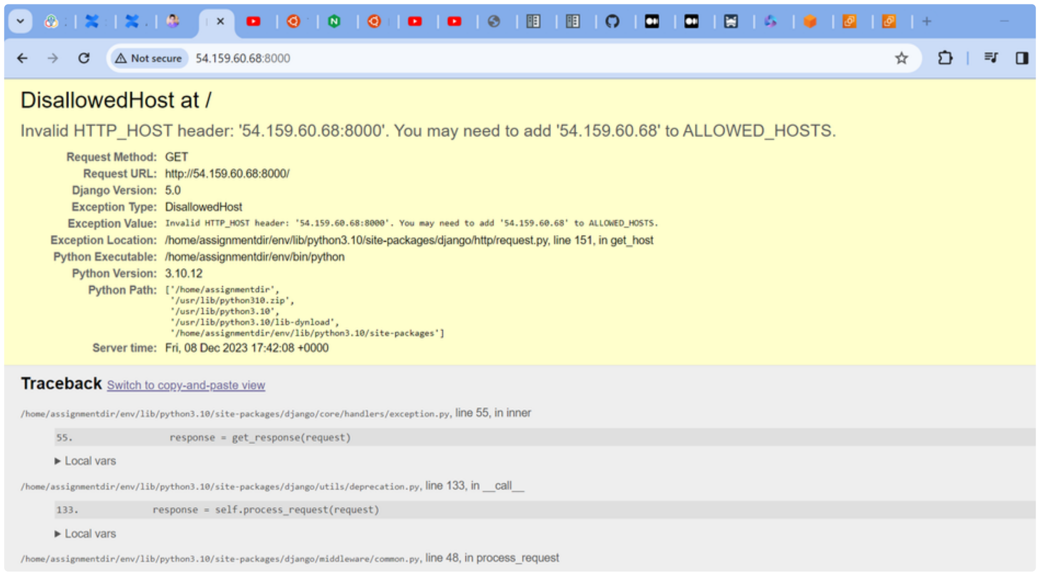
- **manage.py makemigrations**: in django it creates the files that are need to migrate, This command is used to create new database migration files based on the changes you've made to your models.

  When you make changes to your Django models (e.g., adding a new model, adding a field to an existing model, etc.), Django doesn't automatically update the database schema. Instead, it generates a migration file that describes the changes to be made to the database.
- This command analyzes the current state of your models and compares it to the previous state recorded in the previous migrations. It then creates new migration files in the `migrations` directory of each app, specifying the changes needed to bring the database schema up to date.
- **manage.py migrate:** This command is used to apply the changes specified in the migration files to the actual database. It synchronizes the database schema with the current state of your models.Running migrate looks at the migrations directory of each app and applies any pending migrations to the database. It updates the schema and creates or modifies database tables as necessary.

  Note that you should run makemigrations whenever you make changes to your models, and then run migrate to apply those changes to the database. These commands ensure that your database schema stays in sync with your Django models.
- python manage.py runserver - used to simply start a development server, here the host is 127.0.0.1:8000, we need to change it to 0.0.0.0:8000


root@ip-172-31-23-73: /home/assignmentdir

```
Downloading packaging-23.2-py3-none-any.whl (53 kB)
   ---------------------------------------- 53.0/53.0 kB 4.5 MB/s eta 0:00:00
Downloading typing_extensions-4.8.0-py3-none-any.whl (31 kB)
Installing collected packages: typing-extensions, sqlparse, packaging, gunicorn,
 asgiref, django
Successfully installed asgiref-3.7.2 django-5.0 gunicorn-21.2.0 packaging-23.2 s
qlparse-0.4.4 typing-extensions-4.8.0
(env) root@ip-172-31-23-73:/home/assignmentdir# django-admin startproject assign
utils /home/assignmentdir
(env) root@ip-172-31-23-73:/home/assignmentdir# ls
assignutils  env  manage.py
(env) root@ip-172-31-23-73:/home/assignmentdir# python manage.py makemigrations
No changes detected
(env) root@ip-172-31-23-73:/home/assignmentdir# python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
(env) root@ip-172-31-23-73:/home/assignmentdir# sudo ufw allow 8000
Rules updated
Rules updated (v6)
(env) root@ip-172-31-23-73:/home/assignmentdir# python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
December 08, 2023 - 17:37:42
Django version 5.0, using settings 'assignutils.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```



DisallowedHost at /

Invalid HTTP_HOST header: '54.159.60.68:8000'. You may need to add '54.159.60.68' to ALLOWED_HOSTS.

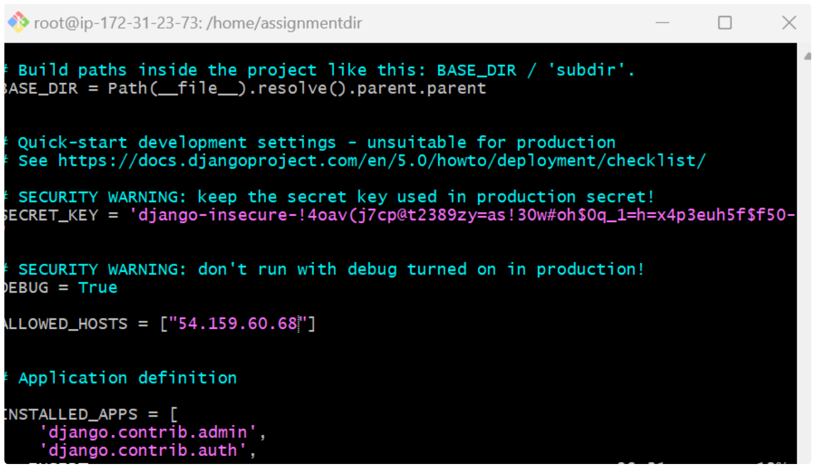
Request Method:	GET
Request URL:	http://54.159.60.68:8000/
Django Version:	5.0
Exception Type:	DisallowedHost
Exception Value:	Invalid HTTP_HOST header: '54.159.60.68:8000'. You may need to add '54.159.60.68' to ALLOWED_HOSTS.
Exception Location:	/home/assignmentdir/env/lib/python3.10/site-packages/django/http/request.py, line 151, in get_host
Python Executable:	/home/assignmentdir/env/bin/python
Python Version:	3.10.12
Python Path:	['/home/assignmentdir', '/usr/lib/python310.zip', '/usr/lib/python3.10', '/usr/lib/python3.10/lib-dynload', '/home/assignmentdir/env/lib/python3.10/site-packages']
Server time:	Fri, 08 Dec 2023 17:42:08 +0000

Traceback Switch to copy-and-paste view

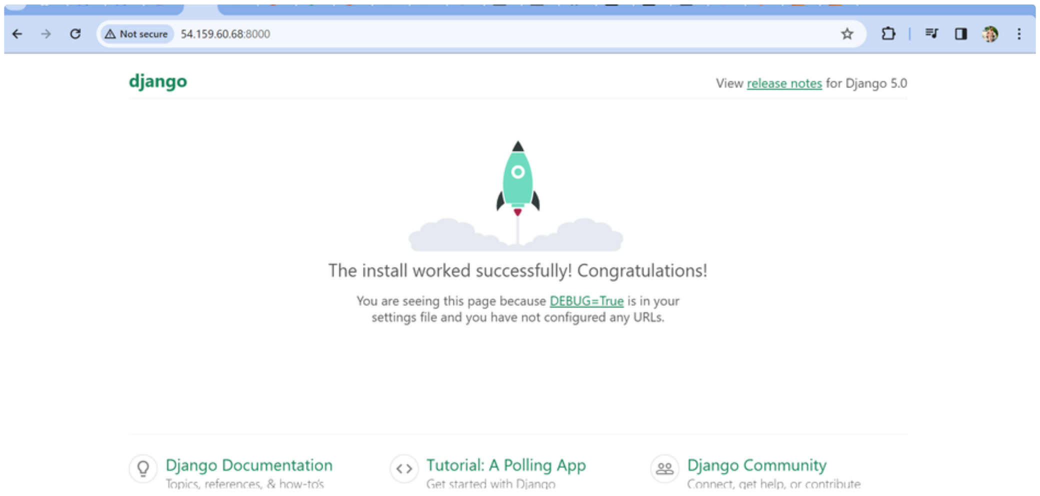
```
/home/assignmentdir/env/lib/python3.10/site-packages/django/core/handlers/exception.py, line 55, in inner
    55.             response = get_response(request)
    ▶ Local vars
/home/assignmentdir/env/lib/python3.10/site-packages/django/utils/deprecation.py, line 133, in __call__
    133.         response = self.process_request(request)
    ▶ Local vars
/home/assignmentdir/env/lib/python3.10/site-packages/django/middleware/common.py, line 48, in process_request
```

- It indicating that you are not allowed to access the django application without adding your IP address to allowed hosts.

```
vim assignutils/settings.py
add 54.159.60.68 to allowed hosts
python manage.py runserver 0.0.0.0:8000
```



```
root@ip-172-31-23-73: /home/assignmentdir
# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/5.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-!4oav(j7cp@t2389zy=as!30w#oh$0q_1=h=x4p3euh5f$f50-

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = ["54.159.60.68"]


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
```

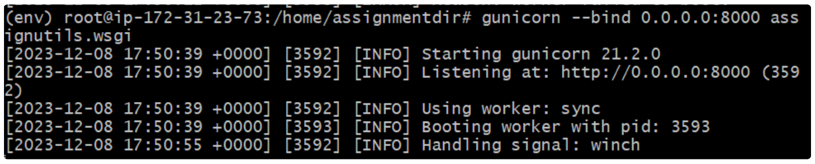


The install worked successfully! Congratulations!

You are seeing this page because DEBUG=True is in your settings file and you have not configured any URLs.

4. **Configuring gunicorn:**

Lets test gunicorn's ability to serve our application. And reload the application page it will be same.



```
(env) root@ip-172-31-23-73:/home/assignmentdir# gunicorn --bind 0.0.0.0:8000 ass
ignutils.wsgi
[2023-12-08 17:50:39 +0000] [3592] [INFO] Starting gunicorn 21.2.0
[2023-12-08 17:50:39 +0000] [3592] [INFO] Listening at: http://0.0.0.0:8000 (359
2)
[2023-12-08 17:50:39 +0000] [3592] [INFO] Using worker: sync
[2023-12-08 17:50:39 +0000] [3593] [INFO] Booting worker with pid: 3593
[2023-12-08 17:50:55 +0000] [3592] [INFO] Handling signal: winch
```

- We want our app to automatically start, and serve as a service.
- Deactivate the virtualenvironment by executing the command below. Let's create a system socket file for gunicorn now. Next, we will create a service file for gunicorn.
- Deactivate the virtual environment after running Gunicorn because Gunicorn is going to be managed as a system service (systemd). **System services typically run in the system's default environment, not within a virtual environment. So, deactivating the virtual environment is a good practice to ensure consistency and avoid potential issues.**
- After deactivating the virtual environment, you proceed to configure Gunicorn as a systemd service and set up Nginx as a reverse proxy. These steps involve system-level configurations, and it's standard to perform them outside the virtual environment.
- This file defines a systemd socket unit for Gunicorn. **Sockets are a way for systemd to listen on a network socket and activate a service when incoming traffic is detected.**

Purpose:

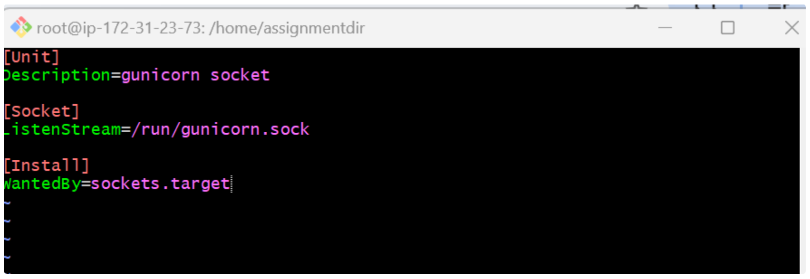
- It sets up a socket that listens for incoming connections on a specified address and port.
- When a connection is established, it activates the corresponding gunicorn.service unit.
- Lets now start and enable the gunicorn socket.

- This file defines a systemd service unit for Gunicorn. Service units are used to specify how a service should be started, stopped, and managed by systemd.

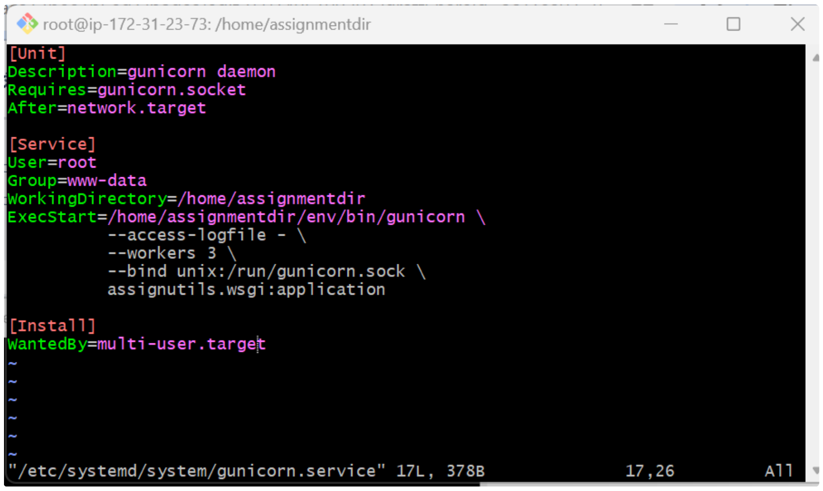
Purpose:

- It sets up the configuration for running Gunicorn as a service.
- Specifies the working directory, user, group, and other parameters.

```
gunicorn --bind 0.0.0.0:8000 assignutils.wsgi
deactivate
sudo vim /etc/systemd/system/gunicorn.socket

[Unit]
Description=gunicorn socket

[Socket]
ListenStream=/run/gunicorn.sock

[Install]
WantedBy=sockets.target

sudo vim /etc/systemd/system/gunicorn.service
[Unit]
Description=gunicorn daemon
Requires=gunicorn.socket
After=network.target
[Service]
User=root
Group=www-data
WorkingDirectory=/home/assignmentdir
ExecStart=/home/assignmentdir/env/bin/gunicorn \
          --access-logfile - \
          --workers 3 \
          --bind unix:/run/gunicorn.sock \
          assignutils.wsgi:application

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



```
[Unit]
Description=gunicorn socket

[Socket]
ListenStream=/run/gunicorn.sock

[Install]
WantedBy=sockets.target
~
~
~
~
```



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

[Service]
User=root
Group=www-data
WorkingDirectory=/home/assignmentdir
ExecStart=/home/assignmentdir/env/bin/gunicorn \
          --access-logfile - \
          --workers 3 \
          --bind unix:/run/gunicorn.sock \
          assignutils.wsgi:application

[Install]
WantedBy=multi-user.target
~
~
~
~
~
~
"/etc/systemd/system/gunicorn.service" 17L, 378B                17,26          All
```

```
sudo systemctl start gunicorn.socket
sudo systemctl enable gunicorn.socket

```

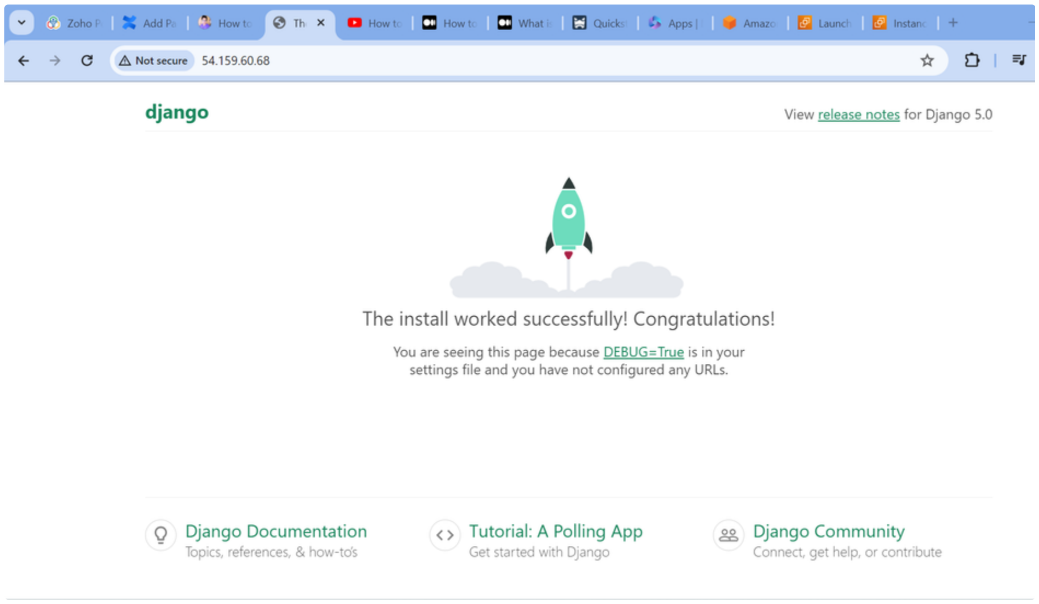
6. **Configuring Nginx as a reverse proxy:**

```
sudo vim /etc/nginx/sites-available/assignutils

server {
    listen 80;
    server_name 54.159.60.68;

    location = /favicon.ico { access_log off; log_not_found off; }
    location /static/ {
        root /home/assignmentdir;
    }

    location / {
        include proxy_params;
        proxy_pass http://unix:/run/gunicorn.sock;
    }
}
```

- Activate the configuration using the following command, by this the content from the config file is moved to sites enabled. Restart nginx and allow the changes to take place.
- Here ln -s is used to create a symbolic link from already created configuration file sites-available to sites-enabled.
- `listen 80;` : Specifies that Nginx should listen on port 80 for incoming HTTP requests.
- `server_name proxy.hemanth.tech;` : Sets the server name. This server block will only be used for requests that match the specified domain name.
- `location = /favicon.ico { ... }` : Configures how Nginx handles requests for the favicon.ico file. In this case, it turns off access log recording and log not found errors for this specific file.
- `location /static/ { ... }` : Defines how Nginx handles requests for files in the `/static/` directory. It sets the root directory for these files to /root/taskdir/.
- `location / { ... }` : This block is for general requests that do not match the previous locations. It includes external proxy parameters and passes the requests to a Unix socket (`http://unix:/run/gunicorn.sock`). This is often used in conjunction with Gunicorn, a WSGI server for running Python web applications.

```
sudo ln -s /etc/nginx/sites-available/assignutils /etc/nginx/sites-enabled/
sudo systemctl restart nginx
```

- This command essentially creates a symlink named `assignutils` in the `sites-enabled` directory that points to the configuration file in the `sites-available` directory. By doing this, you enable the configuration without copying it, allowing you to easily manage which configurations are active.



The install worked successfully! Congratulations!

You are seeing this page because DEBUG=True is in your settings file and you have not configured any URLs.