Glassfish:

- Install Glassfish server and change HTTP port to 8088.

    - GlassFish Server Open Source Edition is an open-source application server built within the GlassFish community. Oracle GlassFish Server is based on GlassFish Server Open Source Edition. GlassFish Server users benefit from a vibrant community that offers self-support, contributes code and product features, product ideas and feedback, bug reports, and more.

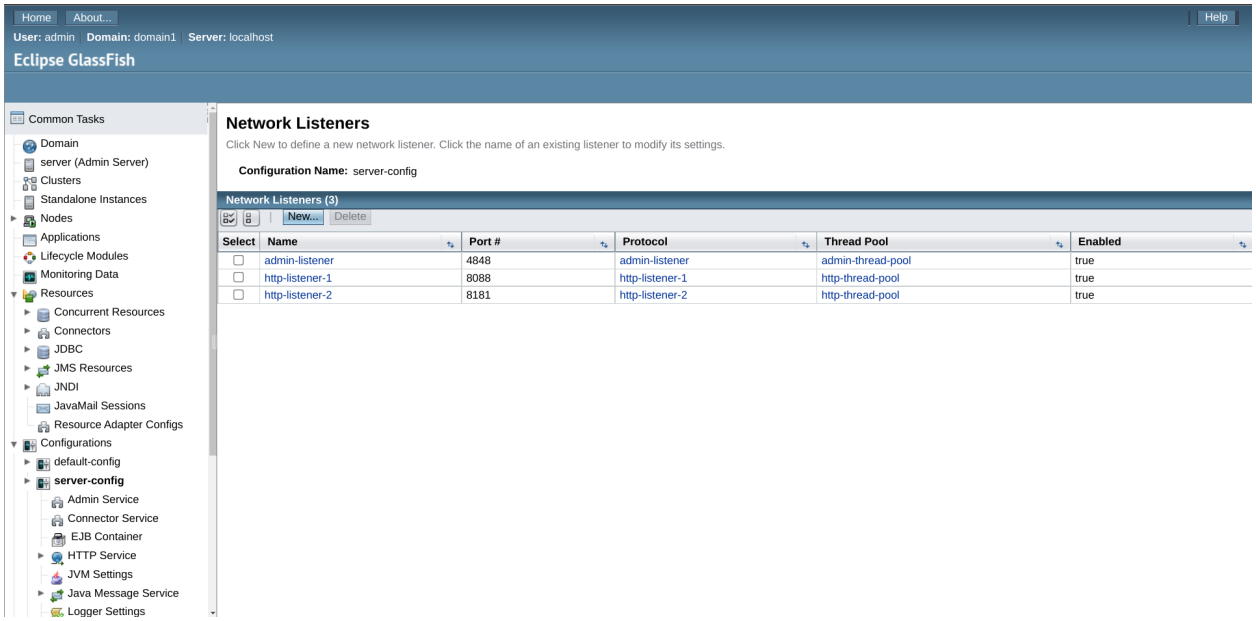
Install GlassFish

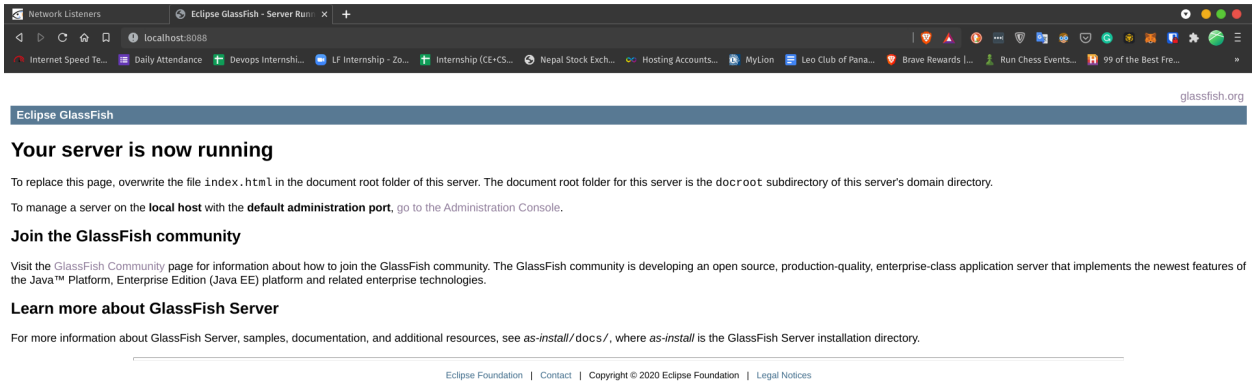
- Install JDK at first using the command:
    - sudo apt install openjdk-11-hre-headless
- Download the zip file for GlassFish 6.2.2 from GitHub link using the command:
    - wget https://github.com/eclipse-ee4j/glassfish/releases/download/6.2.2/glassfish-6.2.2.zip
- Once it's downloaded, extract using the following command:
    - unzip glassfish-6.2.2.zip
- Now we get the directory called glassfish6 inside glassfish-6.2.2.
- Then we can navigate to glassfish4/bin and run the following command to start the server:
- ./asadmin start-domain
- Our server has now been started.

Change HTTP port to 8088

- Run the GlassFish server and connect to the administrative interface using the default port (4848)
- In the left menu go to **Configurations**
- Then select **server-config**
- Now go to **Network Listeners**
- Select **http-listener-1** and change its port value to **8088**

- Click save and restart the server





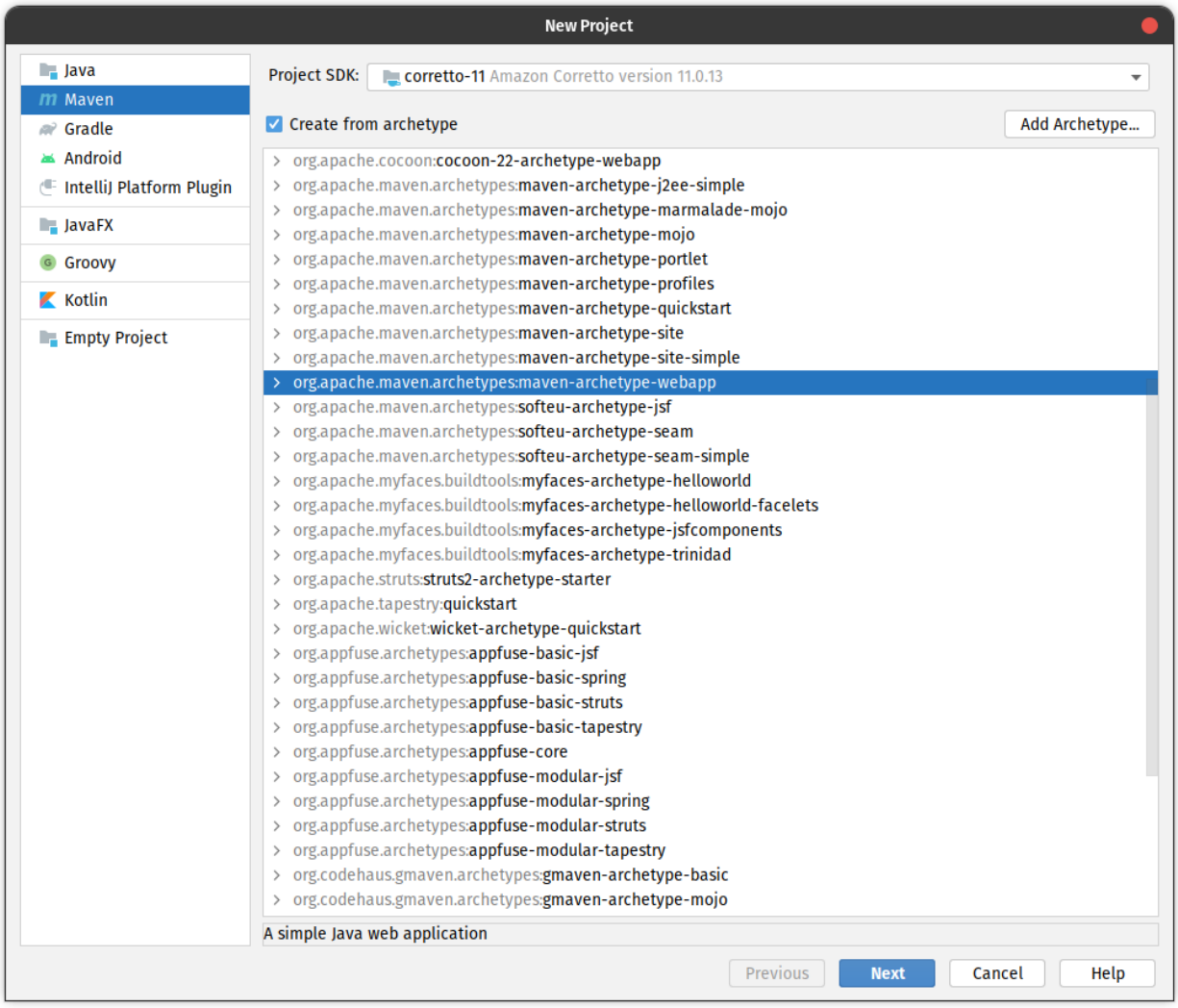
- Create a demo Java (11) servlet application with maven.

  - A Servlet is a java class that is extended to handle the capabilities of a server. Servlets can be used to handle the requests and responses of a server.

- Apache Maven is a software project management and comprehension tool. Maven can help you manage a project's build, reporting, and documentation from a central piece of information.

Demo Java (11) servlet application with maven

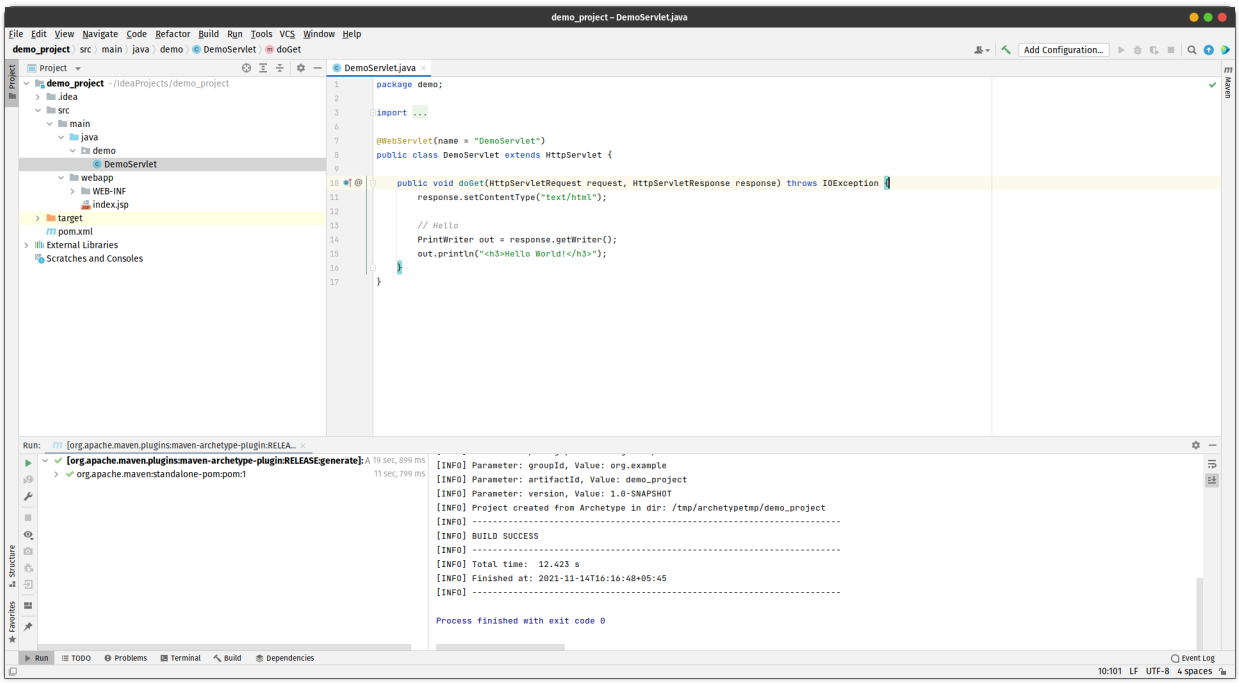
- Install Intellij IDEA and Create a Maven Project



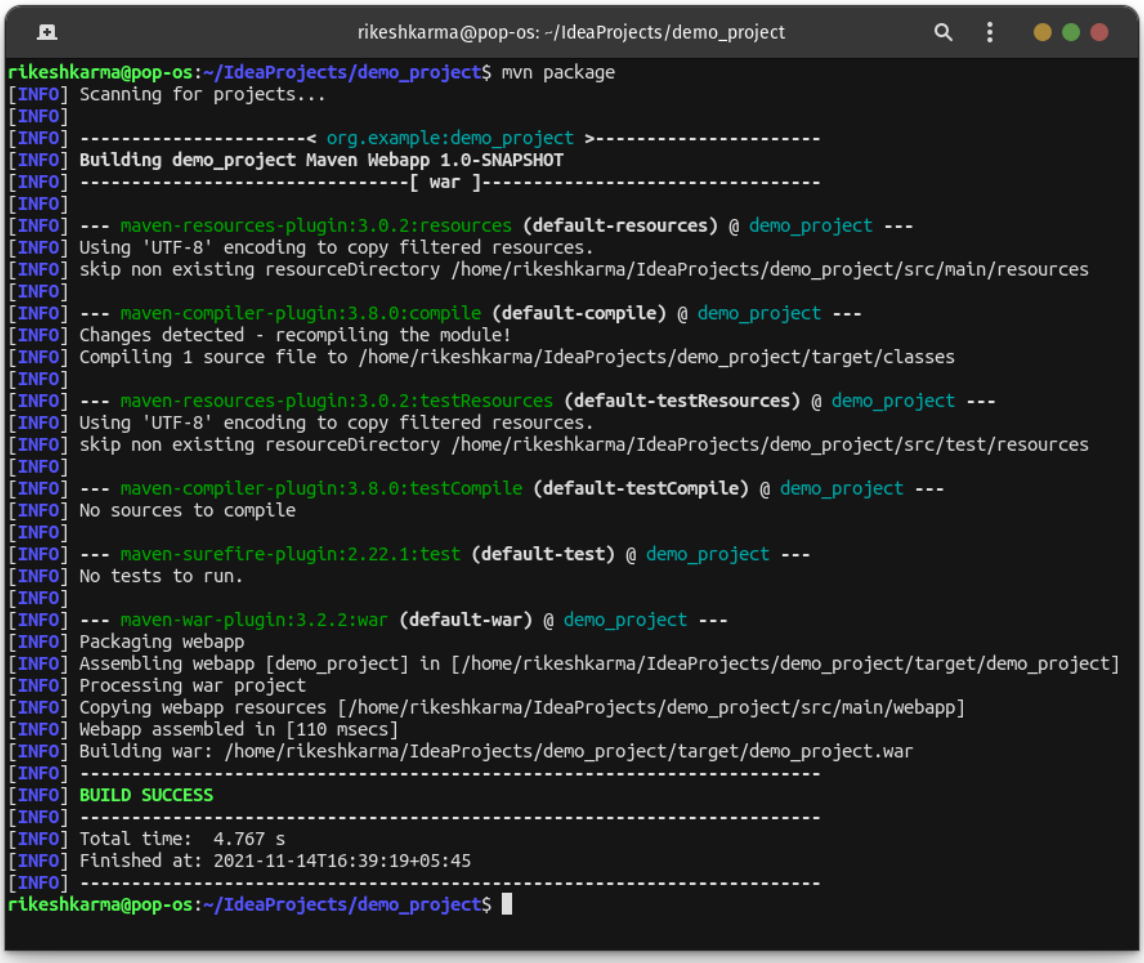
- Using pom.xml to manage the project dependencies
  - POM stands for "Project Object Model". It is an XML representation of a Maven project held in a file named pom.xml.
    - Here we need to add the servlet dependencies in order to work with it.

```
21
22      <dependencies>
23        <dependency>
24          <groupId>junit</groupId>
25          <artifactId>junit</artifactId>
26          <version>4.11</version>
27          <scope>test</scope>
28        </dependency>
29        <!-- https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api -->
30        <dependency>
31          <groupId>javax.servlet</groupId>
32          <artifactId>javax.servlet-api</artifactId>
33          <version>4.0.1</version>
34          <scope>provided</scope>
35        </dependency>
36      </dependencies>
37
```

- Create a new folder named *java* inside src/main folder, then set the java folder to sources root so as to show ‘Class’ on the IntelliJ IDEA when we right-click and select ‘New’.
- Create a new class inside the java folder named *DemoServlet.java* inside the demo folder.

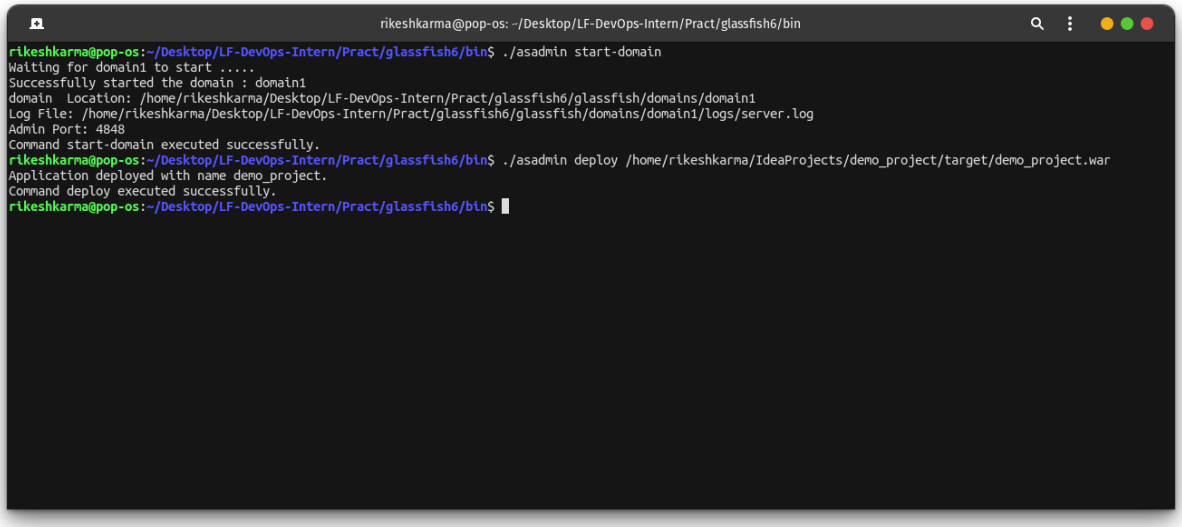


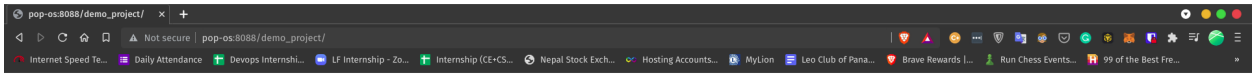
- Generate war package.
    - To generate a war package we need to run the following command inside the project directory:
        - *maven package*
    - The demo_project.war file is generated inside ~/demo_projects/target.



- Deploy the war using the glassfish app server.
    - To deploy we need to start the GlassFish server first by navigating inside glassfish4/bin like before and run the following command to start the server:
        - ./asadmin start-domain
        - Then we need to run the following command to deploy the war file we just generated:

- ./asadmin deploy /home/rikeshkarma/IdeaProjects/demo_project/target/demo_project.war

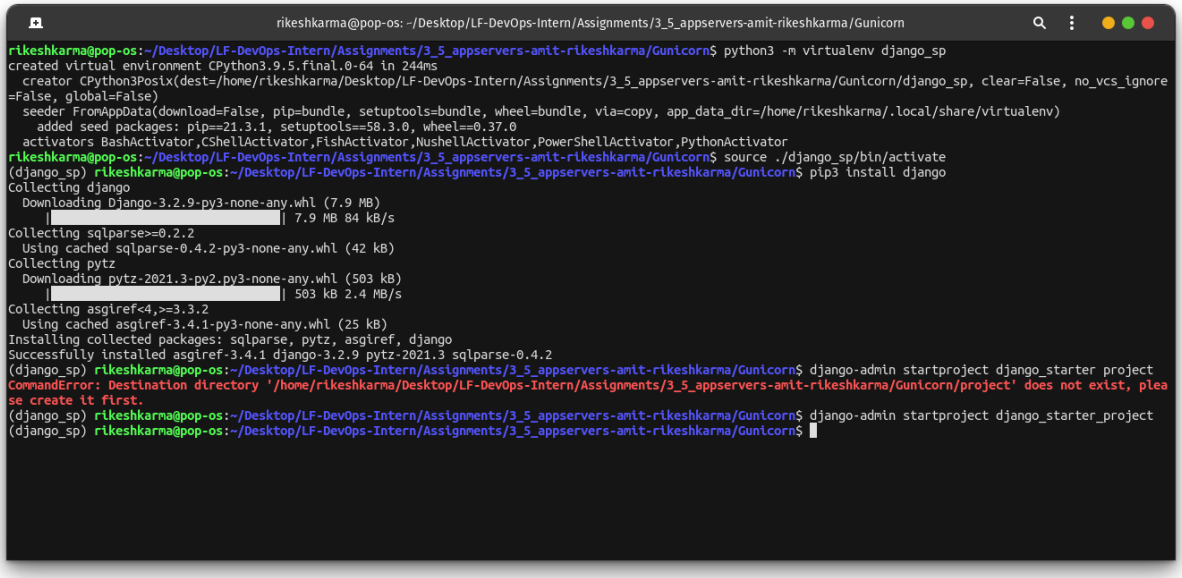




Hello World!

Gunicorn

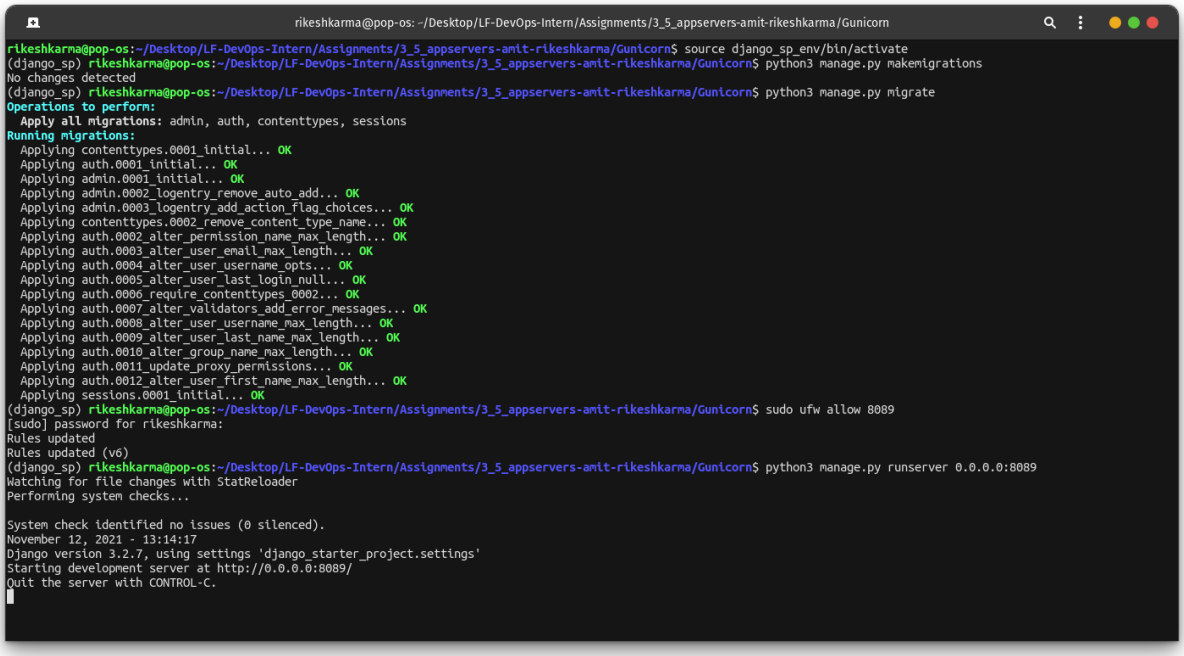
- Create Django starter project in a separate virtual environment.
    - Firstly we install virtualenv using the following commands:
        - pip3 install virtualenv
    - Then create a virtual environment using the command:
        - python3 -m virtualenv django_sp_env
    - Activate the virtual environment using the command:
        - source ./django_sp_env/bin/activate
    - Now we install Django in the virtual environment:
        - pip3 install django
    - Lastly, we create a Django Project, let's call our project **django_starter_project** using the command:
        - django-admin startproject django_starter_project



We have created a Django starter project in a virtual environment.

- Deploy the 3 instances of the application using Gunicorn in 8089 port.
    - Activate the virtual environment and install gunicorn:
        - pip3 install gunicorn
    - Let's add our IP address to the ALLOWED_HOSTS variable in path *django_starter_project/settings.py*

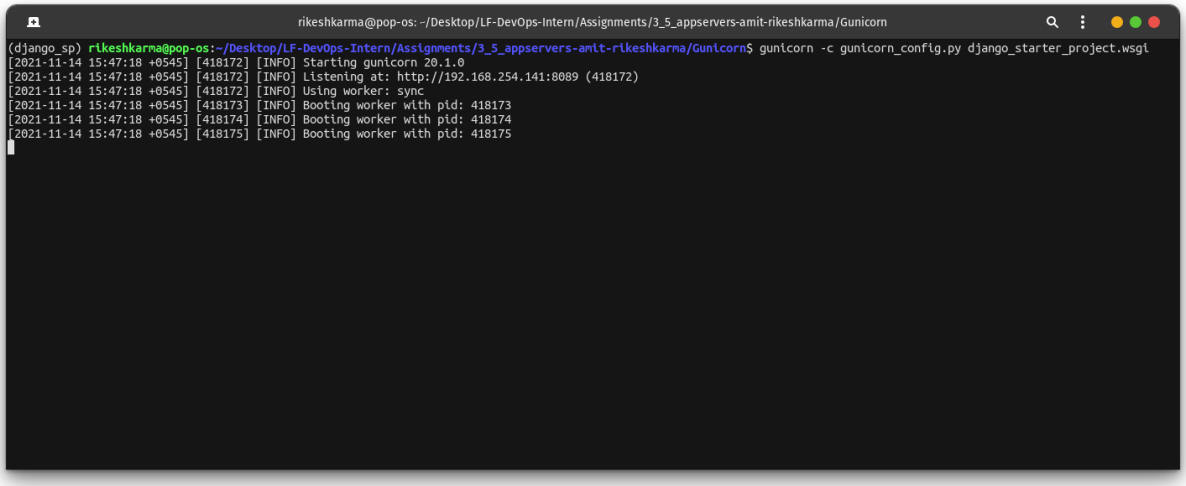
- Now we run migrations:
  - python3 manage.py makemigrations
  - python3 manage.py migrate`
- Then let's test the sample project by running the following command:
  - sudo ufw allow 8089
  - This opens port:8089 by allowing it over the firewall. Let's check our Django server to test the setup so far using the command:
    - python3 manage.py runserver 0.0.0.0:8089

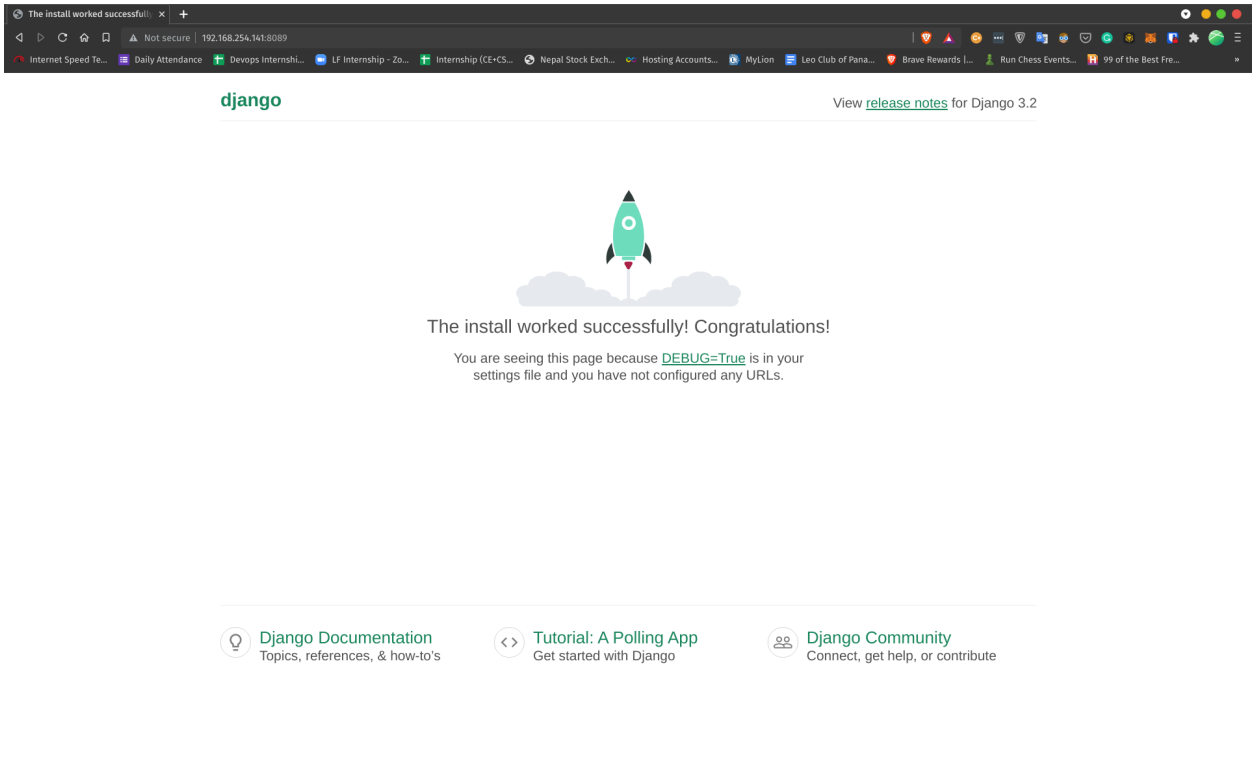


- **Now we configure gunicorn**
  - Firstly, let's make a configuration file for gunicorn server using the following commands:
    - mkdir gunicorn_conf
    - nano config.py

- Then add the following in the file:
    - command='./django_sp_env/bin/gunicorn'
    - pythonPath='./'
    - bind = '192.168.254.141:8089'
    - workers = 3

- After we configure the configuration file we need to run the following command which will deploy 3 instances of the application using Gunicorn in 8089 port:
    - gunicorn -c gunicorn_config.py django_starter_project.wsgi
- Now we can test the server in the browser by using *192.168.254.141:8089*





- Dump access logs in a file in the non-default pattern.
- Dump error logs in a file.
    - We can use following command to dump the log files:
        - *gunicorn app_py:django_starter_project --error-logfile gunicorn.error.log --access-logfile gunicorn.log --capture-output*
        - *And we can* ls to check the files inside
            - We can see ‘gunicorn.error.log’ and ‘gunicorn.log’ files now.