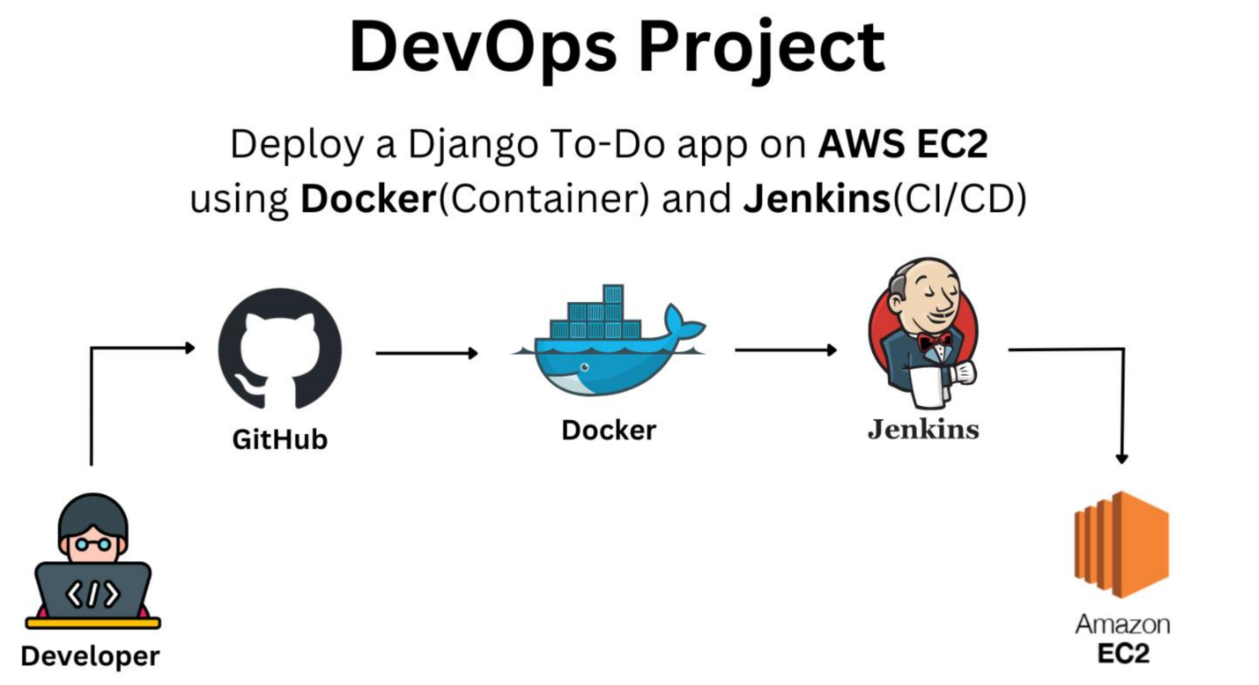
End-to-End CI/CD Pipeline for WebApp using GitHub, Jenkins, Docker, and AWS


DevOps Project

Deploy a Django To-Do app on **AWS EC2** using **Docker**(Container) and **Jenkins**(CI/CD)

Tej Mandaliya

Introduction

Objective:
This project demonstrates the complete DevOps pipeline for a Django Todo App, integrating AWS, Jenkins, and GitHub. The app is fully containerized using Docker and the deployment process is automated with Jenkins CI/CD pipeline.

Technologies used:

- Django: Python-based web framework for developing the Todo app.
- Docker: Containerization of the app for easy deployment.
- AWS EC2: Cloud platform for hosting the application.
- Jenkins: Automation server to manage the CI/CD pipeline.
- GitHub: Version control for managing the project source code.

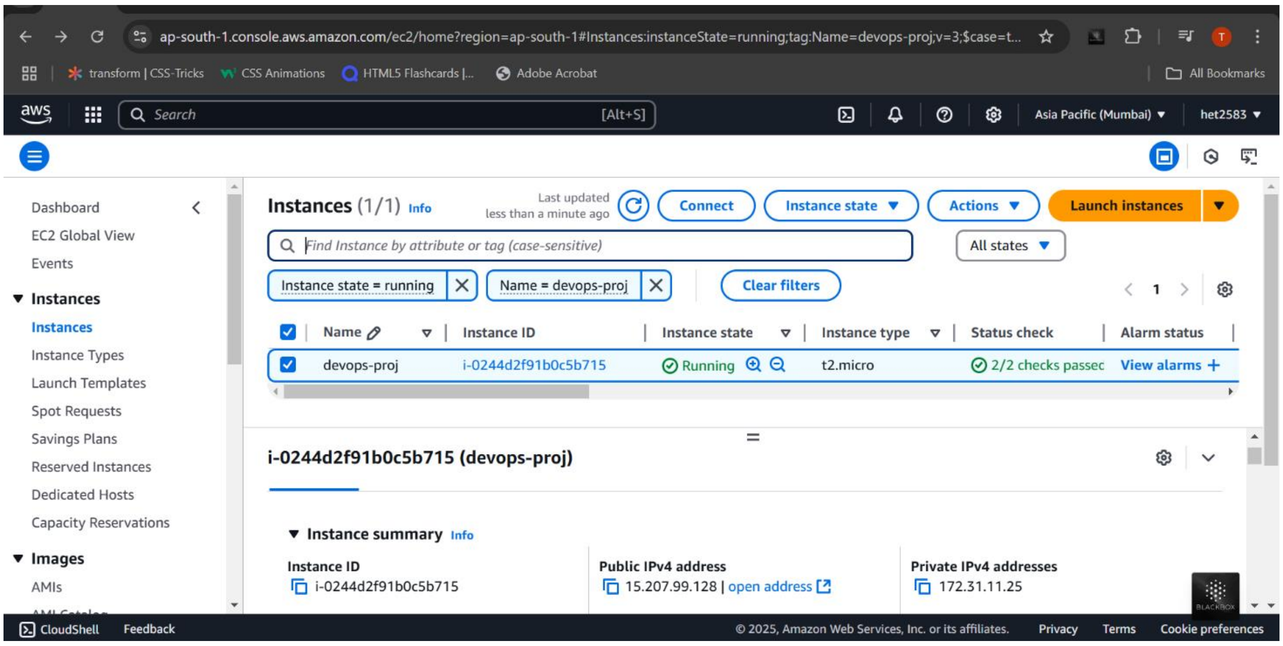
T

Tej Mandaliya

How I Set Up

First Step, I Created AWS EC2 Instance .

- **Launch an EC2 instance (Ubuntu 22.04 or Amazon Linux).**
- **Ensure security group allows inbound traffic for :**
- **SSH (Port 22)**
- **HTTP (Port 80)**
- **Custom TCP Rule (Port 8000 for Django project)**
  - **Connect to your instance via SSH.**



Instances (1/1) Info

Find Instance by attribute or tag (case-sensitive)

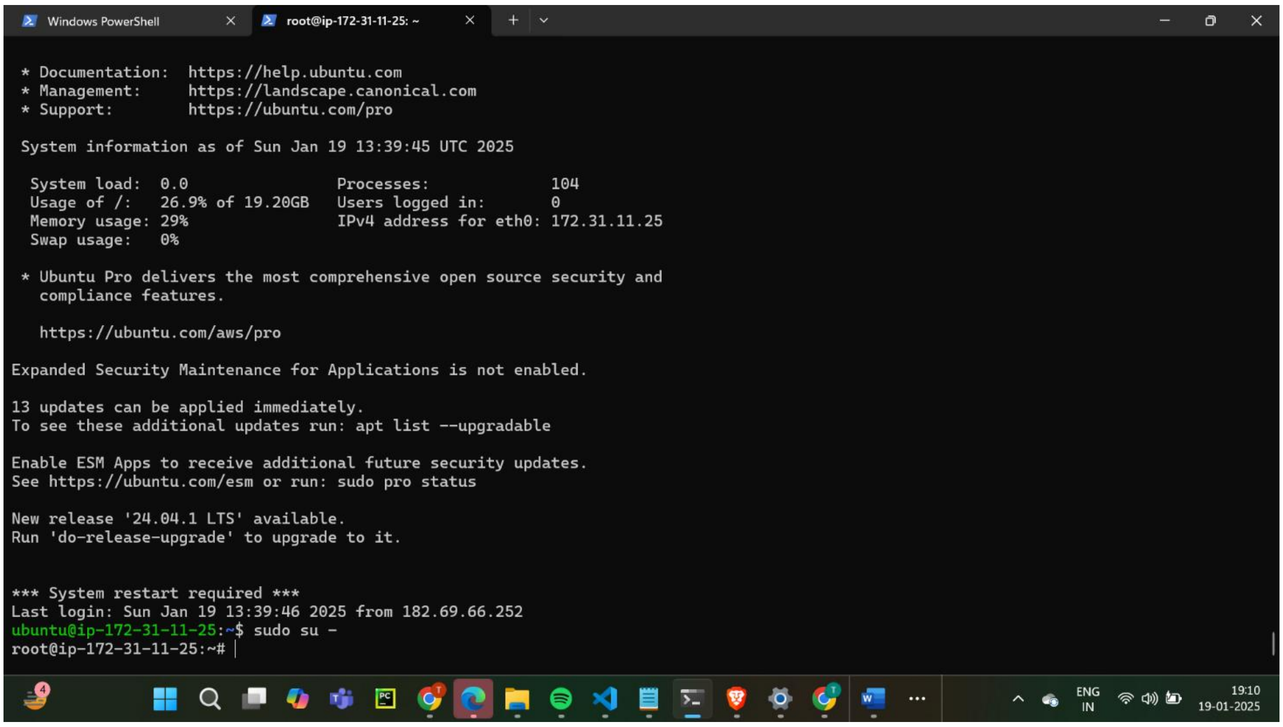
Instance state = running | Name = devops-proj

Name	Instance ID	Instance state	Instance type	Status check	Alarm status
devops-proj	i-0244d2f91b0c5b715	Running	t2.micro	2/2 checks passec	View alarms +

i-0244d2f91b0c5b715 (devops-proj)

Instance summary Info

Instance ID	Public IPv4 address	Private IPv4 addresses
i-0244d2f91b0c5b715	15.207.99.128 | open address	172.31.11.25



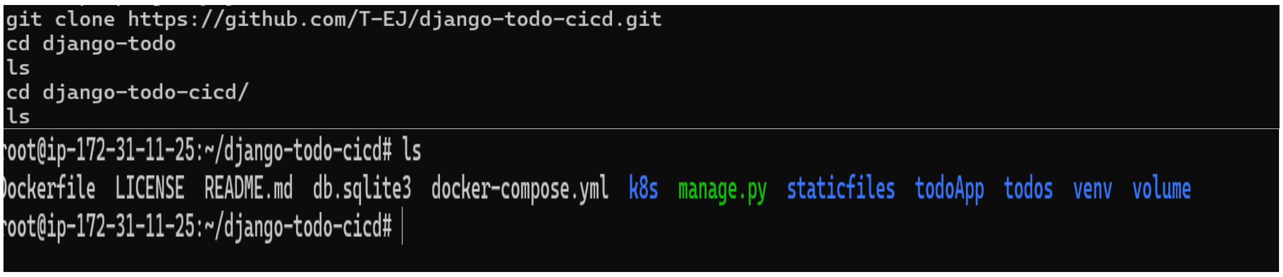
```
 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Sun Jan 19 13:39:45 UTC 2025

  System load:  0.0               Processes:             104
  Usage of /:   26.9% of 19.20GB   Users logged in:       0
  Memory usage: 29%               IPv4 address for eth0: 172.31.11.25
  Swap usage:   0%

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

13 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

New release '24.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


*** System restart required ***
Last login: Sun Jan 19 13:39:46 2025 from 182.69.66.252
ubuntu@ip-172-31-11-25:~$ sudo su -
root@ip-172-31-11-25:~#
```

Second Step **Install Docker, Docker, Git, Python3, pip3, and Django.**

- **Sudo apt update**
- **sudo apt install -y docker.io**
- **sudo apt install -y git**
- **sudo apt install -y python3 python3-pip**
- **pip3 install Django**

Third Step: Clone the Project

Clone the project from GitHub into the EC2 instance.

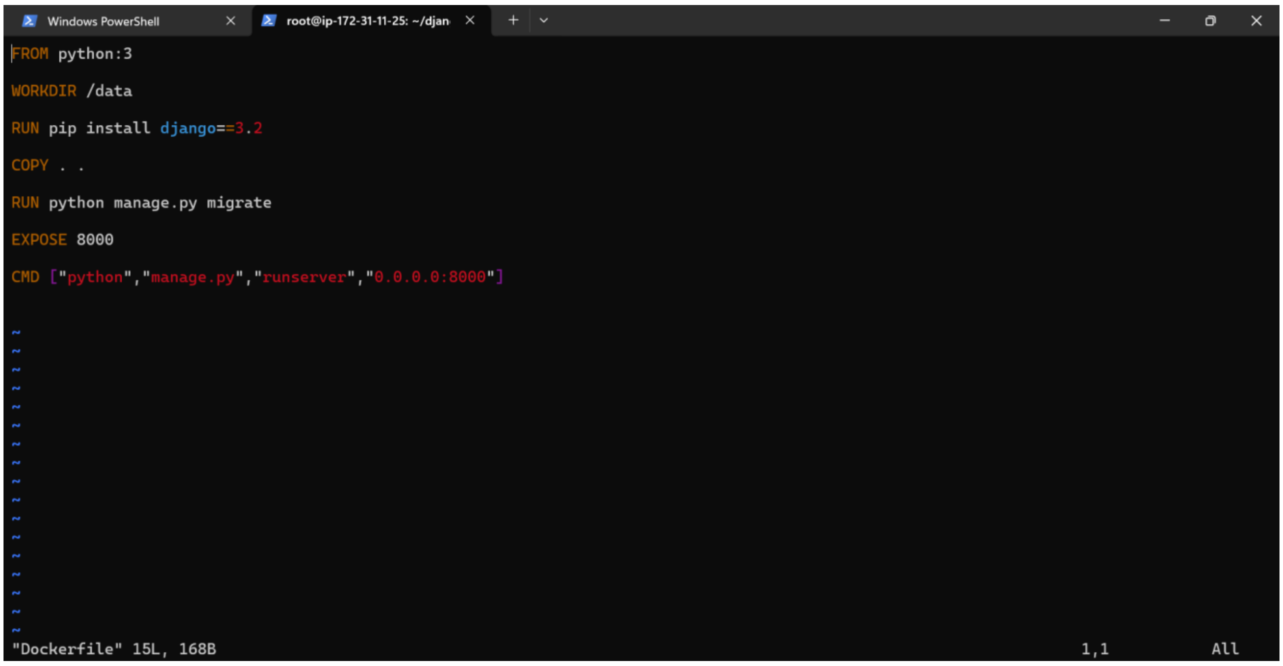


```
git clone https://github.com/T-EJ/django-todo-cicd.git
cd django-todo
ls
cd django-todo-cicd/
ls
```

```
oot@ip-172-31-11-25:~/django-todo-cicd# ls
ockerfile  LICENSE  README.md  db.sqlite3  docker-compose.yml  k8s  manage.py  staticfiles  todoApp  todos  venv  volume
oot@ip-172-31-11-25:~/django-todo-cicd#
```

Fourth Step:

To containerize my Django application, I created a Dockerfile. This Dockerfile defines the environment for my application and specifies the steps to build the image.



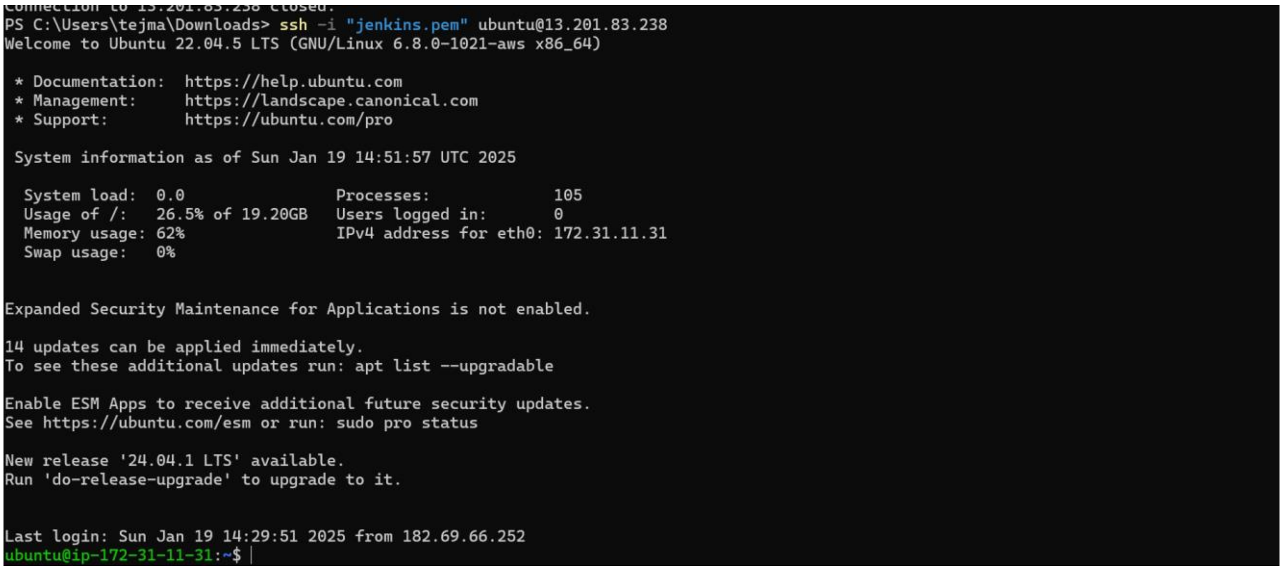
```
FROM python:3

WORKDIR /data

RUN pip install django==3.2

COPY . .

RUN python manage.py migrate

EXPOSE 8000

CMD ["python","manage.py","runserver","0.0.0.0:8000"]

~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"Dockerfile" 15L, 168B                                             1,1          All
```

Fifth Step: Set Up Jenkins for CI/CD

To automate the deployment process, I set up a Jenkins server. While I created a separate instance for Jenkins, you can also configure Jenkins on the same EC2 instance used earlier.

Ensure the security group for this instance allows:

- **Port 8080 (default port for Jenkins) for web access.**
- **Connect to your instance via SSH.**



```
PS C:\Users\tejma\Downloads> ssh -i "jenkins.pem" ubuntu@13.201.83.238
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-1021-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Sun Jan 19 14:51:57 UTC 2025

  System load:  0.0                Processes:             105
  Usage of /:   26.5% of 19.20GB   Users logged in:       0
  Memory usage: 62%                IPv4 address for eth0: 172.31.11.31
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

14 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

New release '24.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Sun Jan 19 14:29:51 2025 from 182.69.66.252
ubuntu@ip-172-31-11-31:~$ |
```

Step 5.1: Update the System

Before installing Jenkins, ensure your system packages are up-to-date:

- **sudo apt update**

Step 5.2: Install Java

Jenkins requires Java to run. Install OpenJDK 11 using the command:

- **sudo apt install -y openjdk-11-jdk**

Step 5.3: Add Jenkins Repository and Import GPG Key

Add the official Jenkins repository to your system and import its GPG key:

- **curl -fsSL https://pkg.jenkins.io/debian/jenkins.io.key | sudo tee \ /usr/share/keyrings/jenkins-keyring.asc > /dev/null**

Add the Jenkins repository to your system's package manager:

- **echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \ https://pkg.jenkins.io/debian binary/ | sudo tee \ /etc/apt/sources.list.d/jenkins.list > /dev/null**

Step 5.4: Update the Package List Again

After adding the Jenkins repository, refresh the package list:

- **sudo apt update**

Step 5.5: Install Jenkins

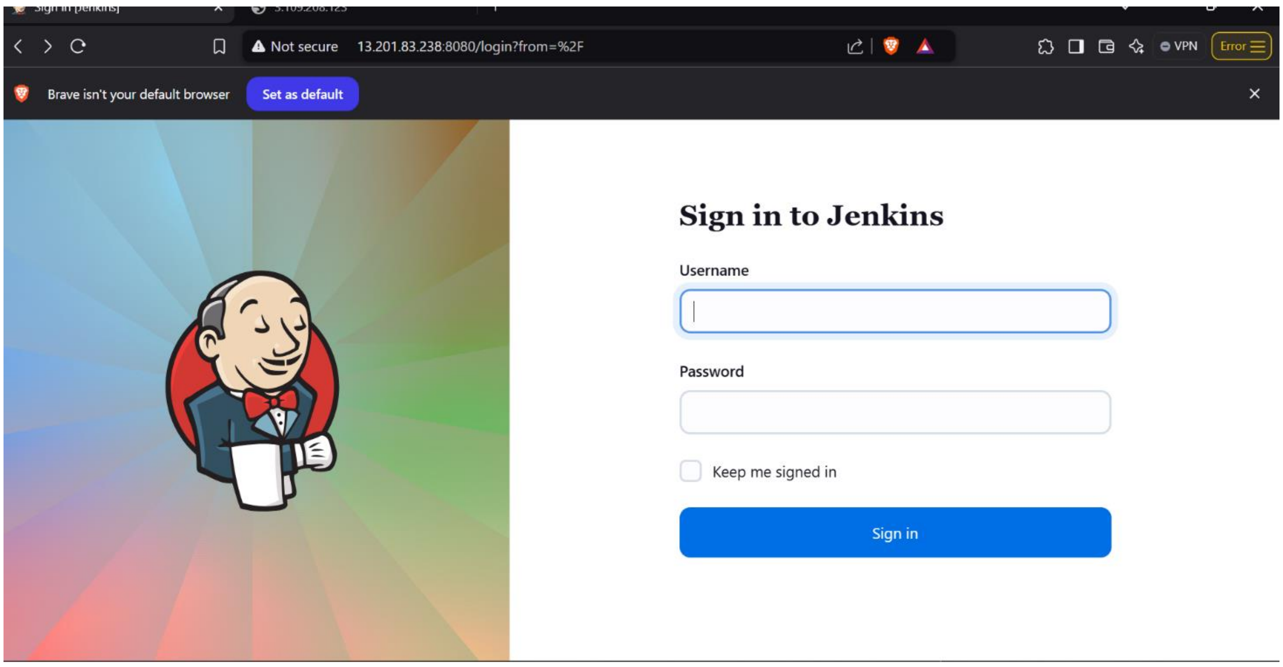
- **Install the Jenkins application:**
- **sudo apt install -y jenkins**

Step 5.6: Start and Enable Jenkins Service

Start the Jenkins service and enable it to automatically start on boot:

- **sudo systemctl start jenkins**

- **sudo systemctl enable Jenkins**
- **Now Start Jenkins server <ip-address>:8080**


Not secure
13.201.83.238:8080/login?from=%2F
VPN
Error
Brave isn't your default browser
Set as default

Sign in to Jenkins

Username

Password

Keep me signed in

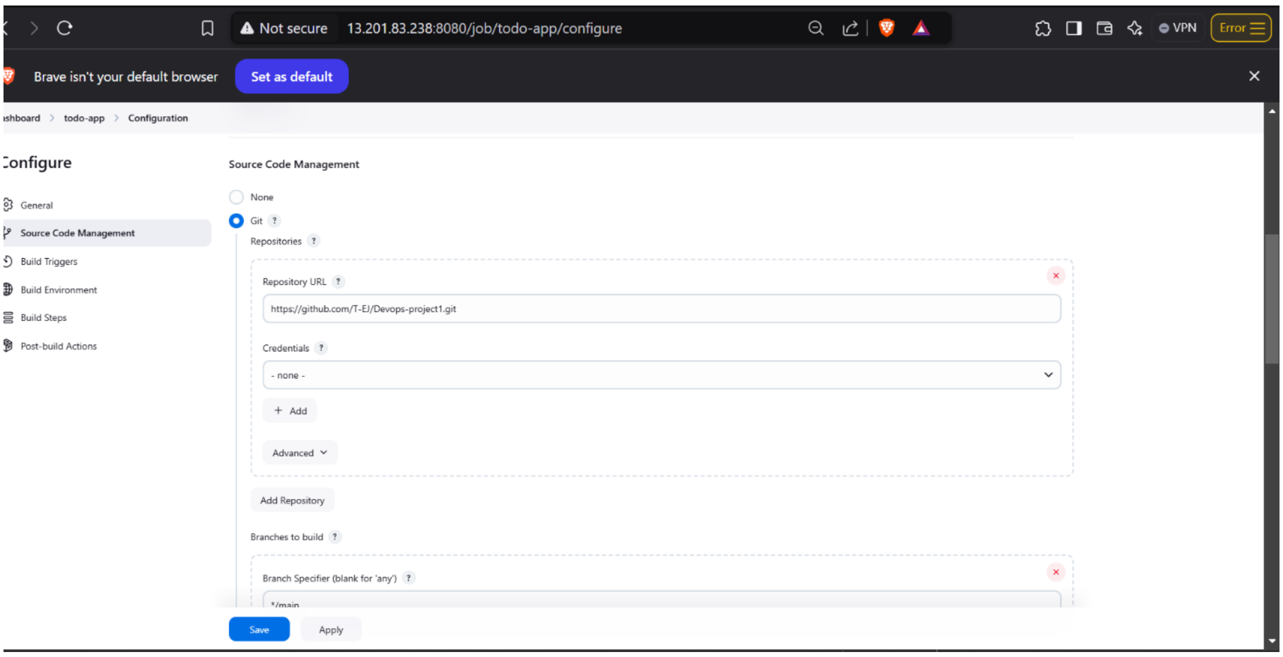
Sign in

Sixth Step: Integrate GitHub with Jenkins

1. Go to **Manage Jenkins** → **System** → **GitHub Server**.
2. Add the GitHub URL and a personal access token.
3. Save the configuration.

Create a Freestyle Job

1. Create a new **Freestyle Job** in Jenkins.
2. Execute a shell build to verify that the project runs correctly.



Source Code Management

None

[✓ Git]

Repositories

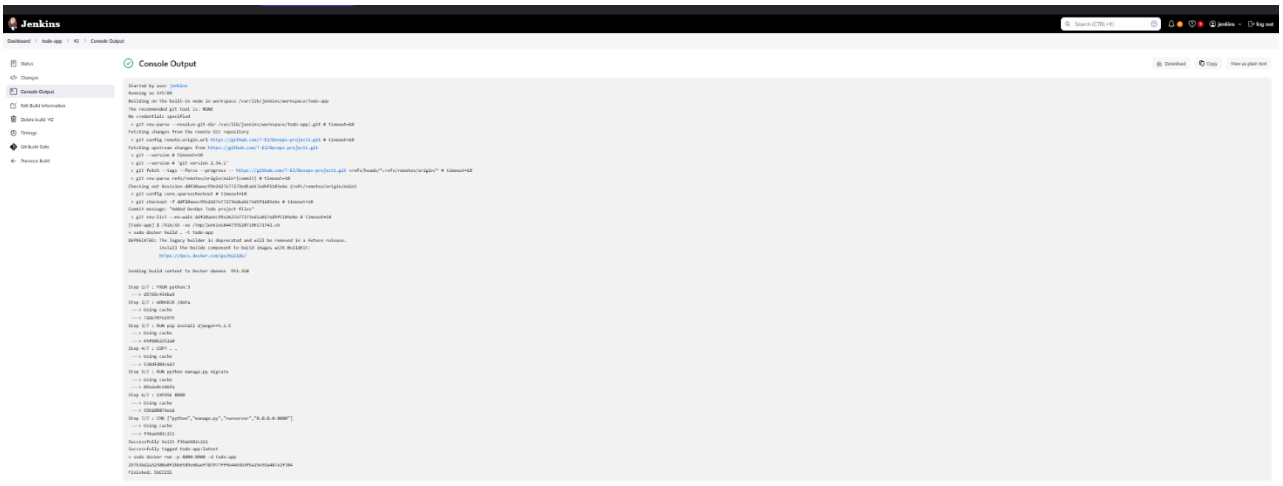
Repository URL: https://github.com/T-EI/Devops-project1.git

Credentials: - none -

Branches to build

Branch Specifier (blank for 'any'): */main

- **Just for Checking I have executed shell to check build is running proper or not**

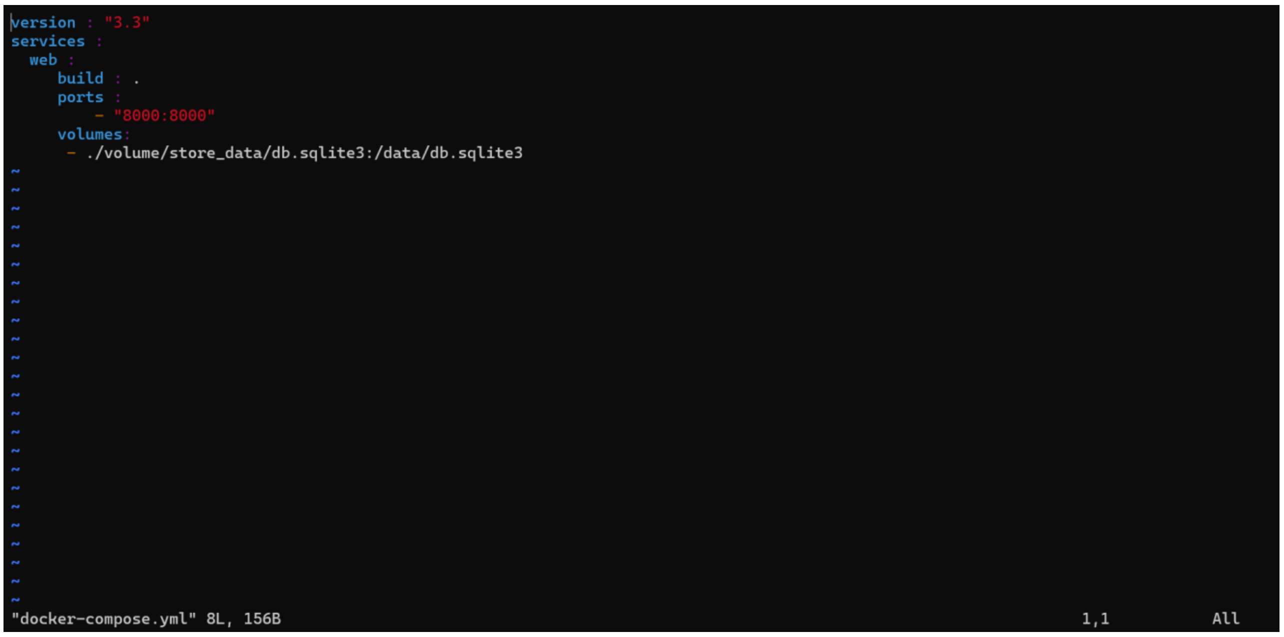


```
Started by user jenkins
Running as SYSTEM
Building on the built-in node in workspace /var/lib/jenkins/workspace/todo-app
The recommended git tool is: NONE
No credentials specified
 > git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/todo-app/.git # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/T-EJ/Devops-project1.git # timeout=10
Fetching upstream changes from https://github.com/T-EJ/Devops-project1.git
 > git --version # timeout=10
 > git --version # 'git version 2.34.1'
 > git fetch --tags --force --progress -- https://github.com/T-EJ/Devops-project1.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision ddf20aeec99e2627e77273ed1a617ed5f1185e6e (refs/remotes/origin/main)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f ddf20aeec99e2627e77273ed1a617ed5f1185e6e # timeout=10
Commit message: "Added DevOps Todo project files"
 > git rev-list --no-walk ddf20aeec99e2627e77273ed1a617ed5f1185e6e # timeout=10
[todo-app] $ /bin/sh -xe /tmp/jenkins8447991207186171742.sh
+ sudo docker build . -t todo-app
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  951.3kB

Step 1/7 : FROM python:3
 ---> d0760c436ba0
Step 2/7 : WORKDIR /data
 ---> Using cache
 ---> 72de707e2935
Step 3/7 : RUN pip install django==5.1.5
 ---> Using cache
 ---> 65f60b2252a0
Step 4/7 : COPY . .
 ---> Using cache
 ---> 538dfd8dc6d3
Step 5/7 : RUN python manage.py migrate
 ---> Using cache
 ---> 00a2e0c106fa
Step 6/7 : EXPOSE 8000
 ---> Using cache
 ---> 5061088 76e16
Step 7/7 : CMD ["python","manage.py","runserver","0.0.0.0:8000"]
 ---> Using cache
 ---> f36aeb82c211
Successfully built f36aeb82c211
Successfully tagged todo-app:latest
+ sudo docker run -p 8000:8000 -d todo-app
297b3bd2a32308e0f1886589e68aef787977fff e4463b3fba15e59a487e2f784
Finished: SUCCESS
```

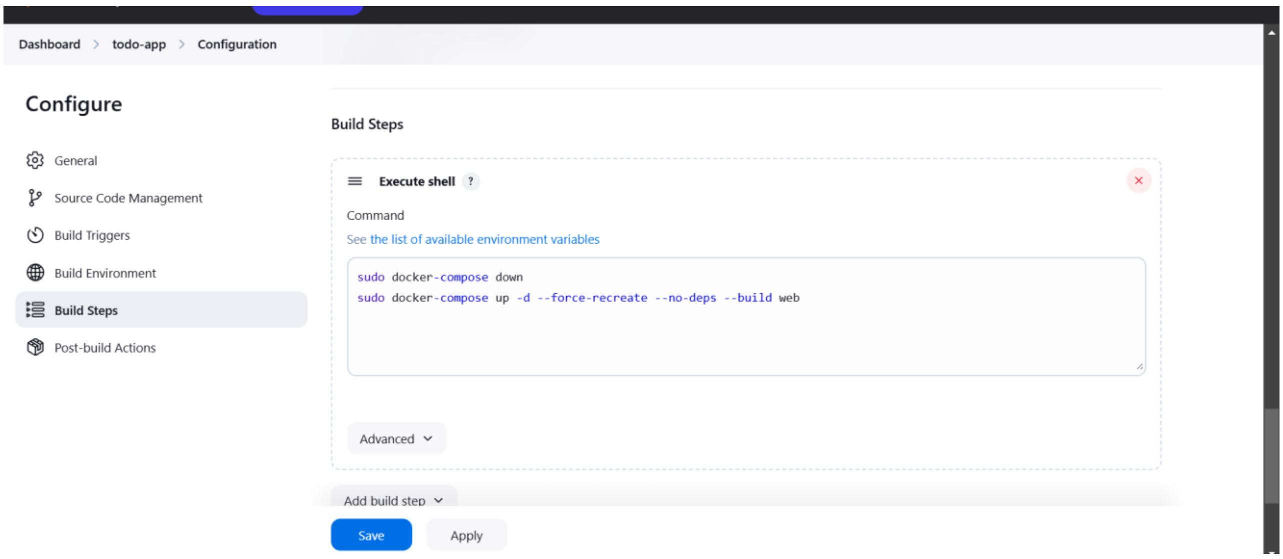
Seventh Step: Automate with Docker Compose

Create a docker-compose.yml file for the Django application. This ensures that:

- **Builds are created automatically.**
- **You don't need to manually stop or restart processes when running builds.**

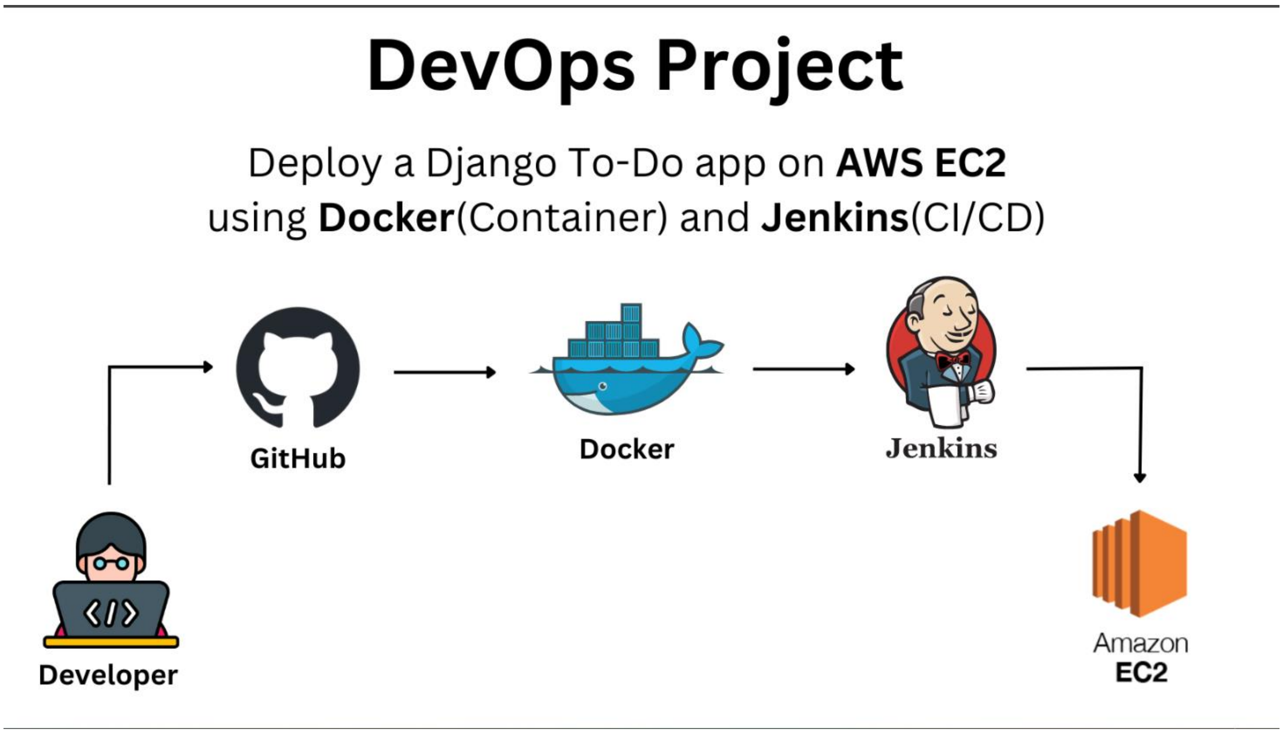


```
version : "3.3"
services :
  web :
     build : .
     ports :
         - "8000:8000"
     volumes:
      - ./volume/store_data/db.sqlite3:/data/db.sqlite3
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"docker-compose.yml" 8L, 156B                                              1,1           All
```



```
sudo docker-compose down
sudo docker-compose up -d --force-recreate --no-deps --build web
```

- **If you find this project helpful or would like to explore it further, please give it a like and download it from the GitHub repository: https://github.com/T-EJ/Devops-project1.git**


DevOps Project
Deploy a Django To-Do app on AWS EC2
using Docker(Container) and Jenkins(CI/CD)
GitHub
Docker
Jenkins
Developer
Amazon EC2