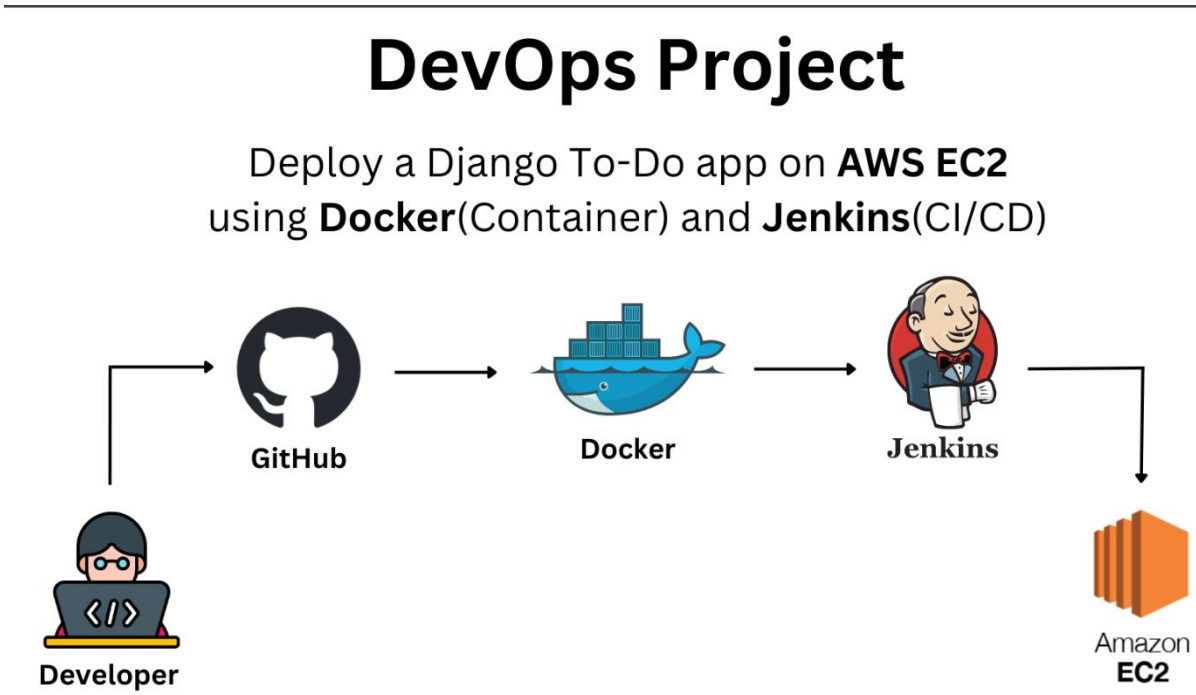
End-to-End CI/CD Pipeline for WebApp using GitHub, Jenkins, Docker, and AWS



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:

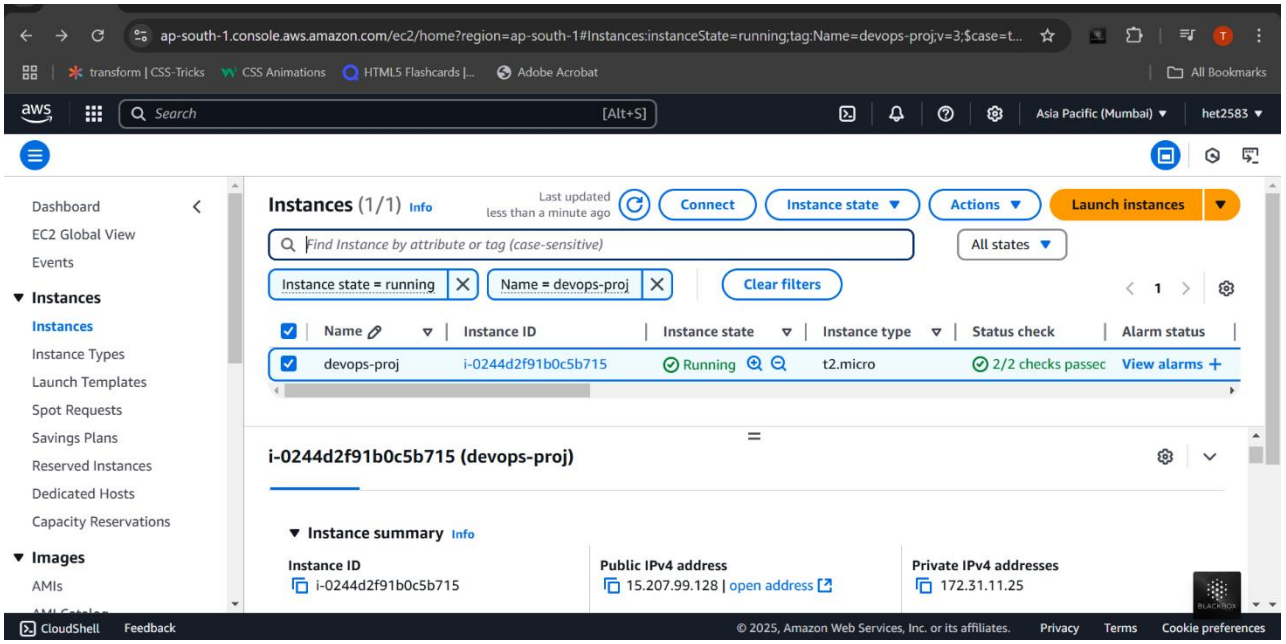
- Django: Python-based web framework for developing the Todo app.
- Docker: Containerization of the app for easy deployment.
- AWS EC2: Cloud platform for hosting the application.
- Jenkins: Automation server to manage the CI/CD pipeline.
- GitHub: Version control for managing the project source code.

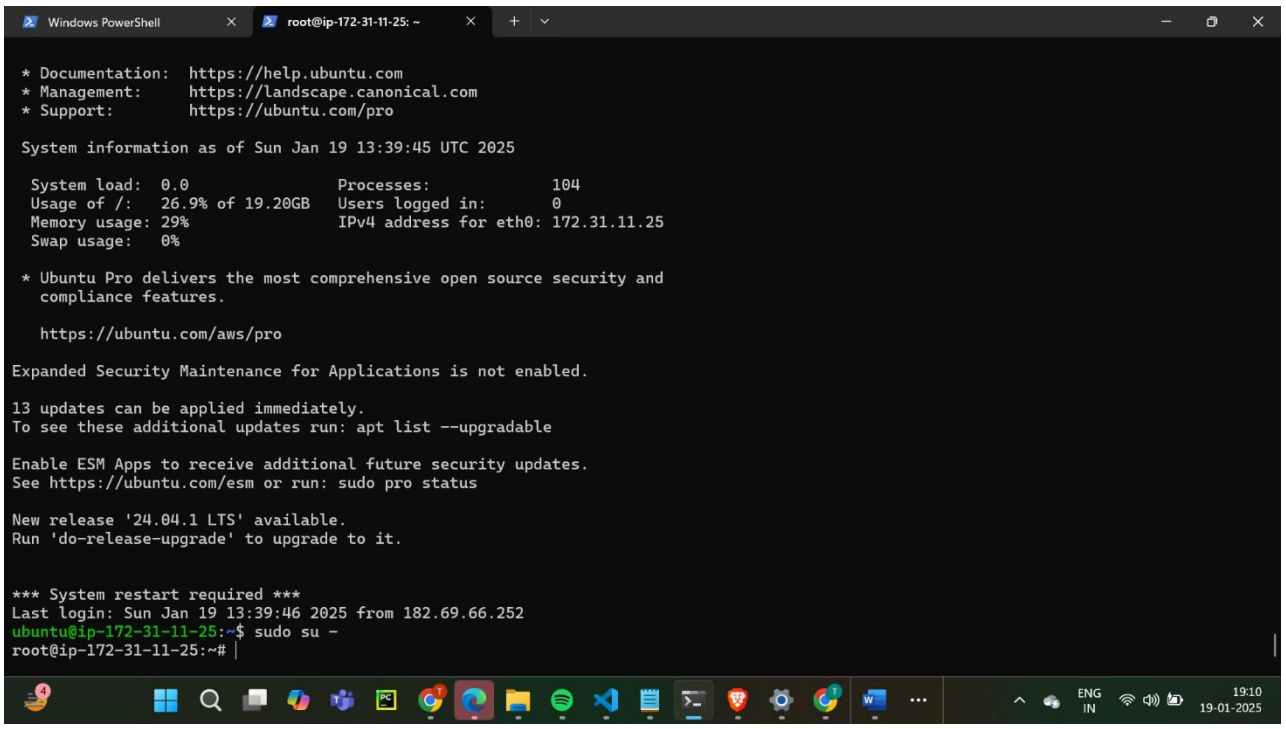
Tej Mandaliya

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.





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

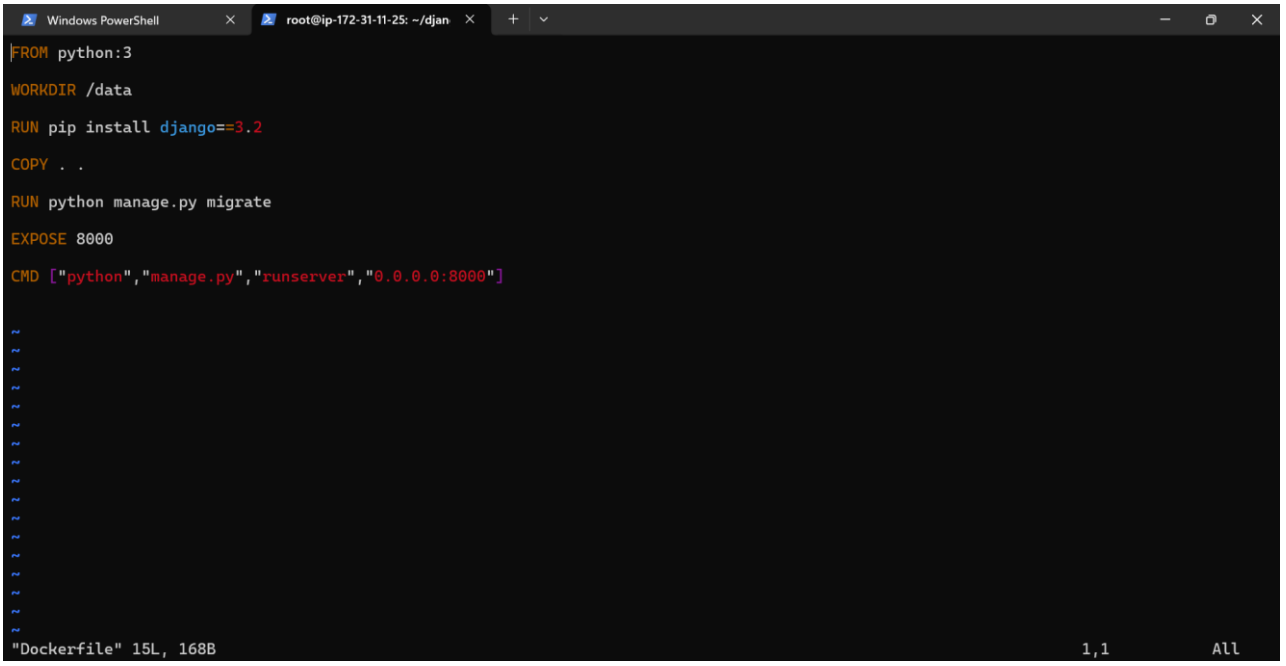
Third Step: Clone the Project

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.

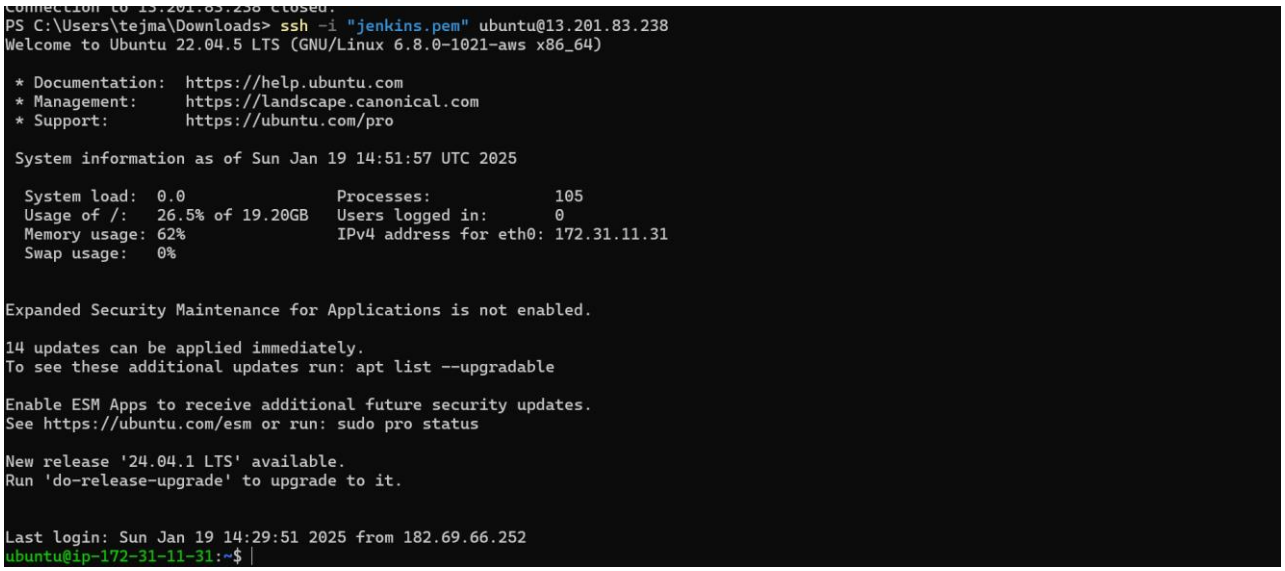


Fifth Step: Set Up Jenkins for CI/CD

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.**



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

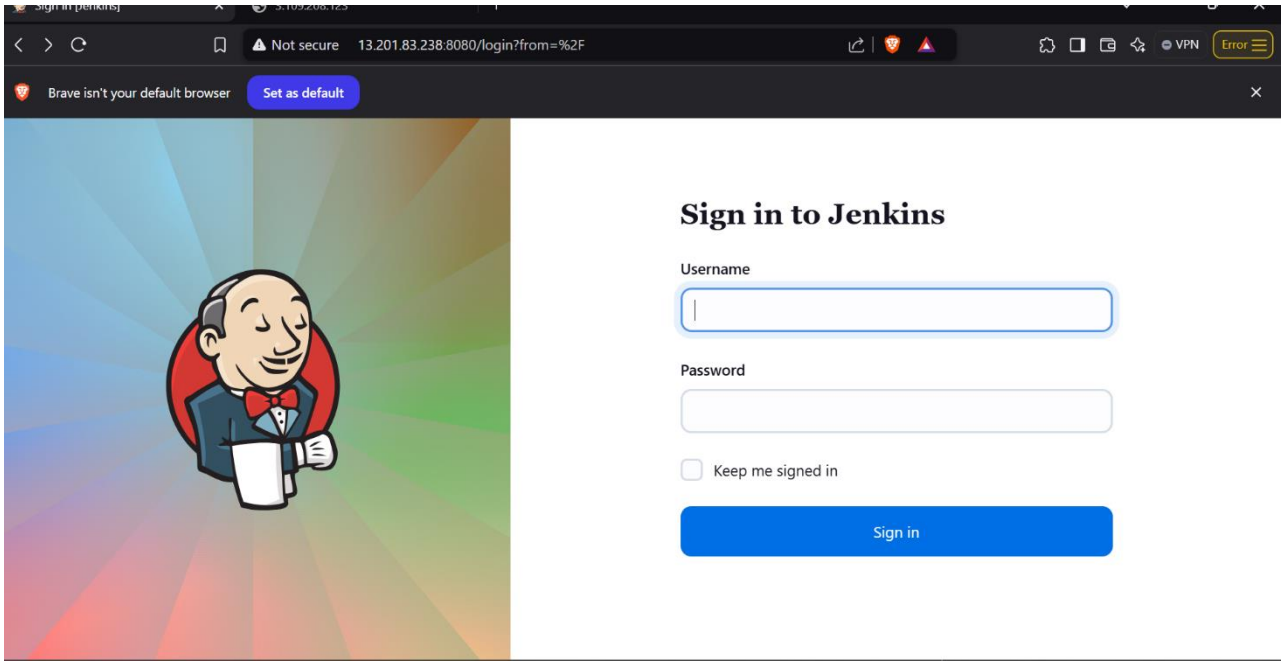
- **Install the Jenkins application:**
- **sudo apt install -y jenkins**

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**

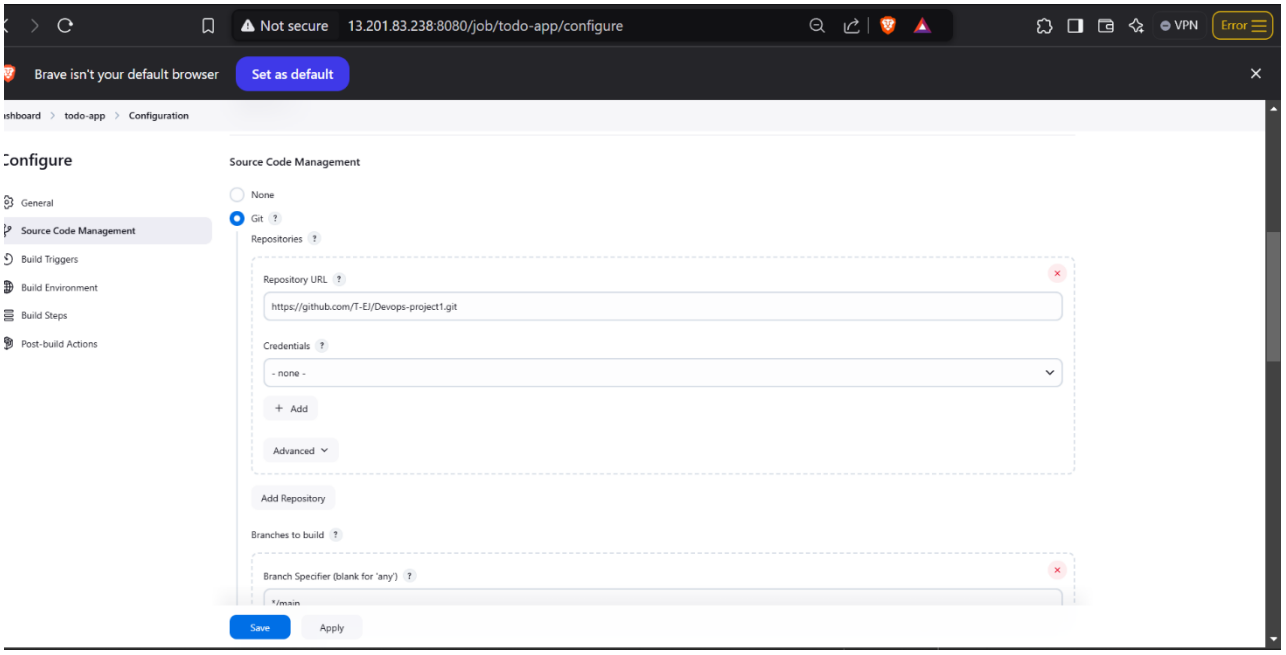


Sixth Step: Integrate GitHub with Jenkins

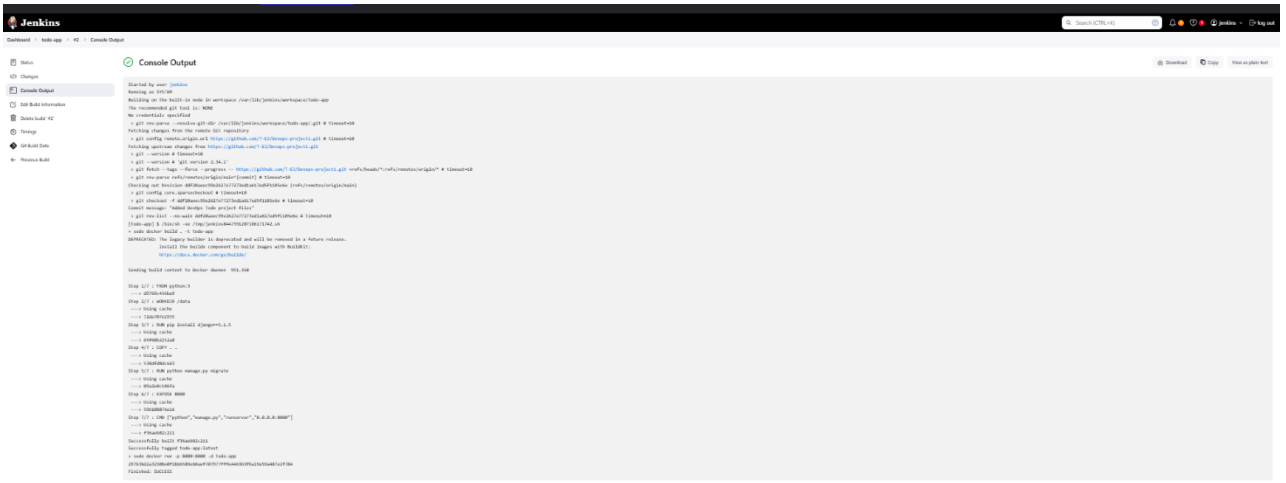
1. Go to **Manage Jenkins** → **System** → **GitHub Server**.
2. Add the GitHub URL and a personal access token.
3. Save the configuration.

Create a Freestyle Job

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



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

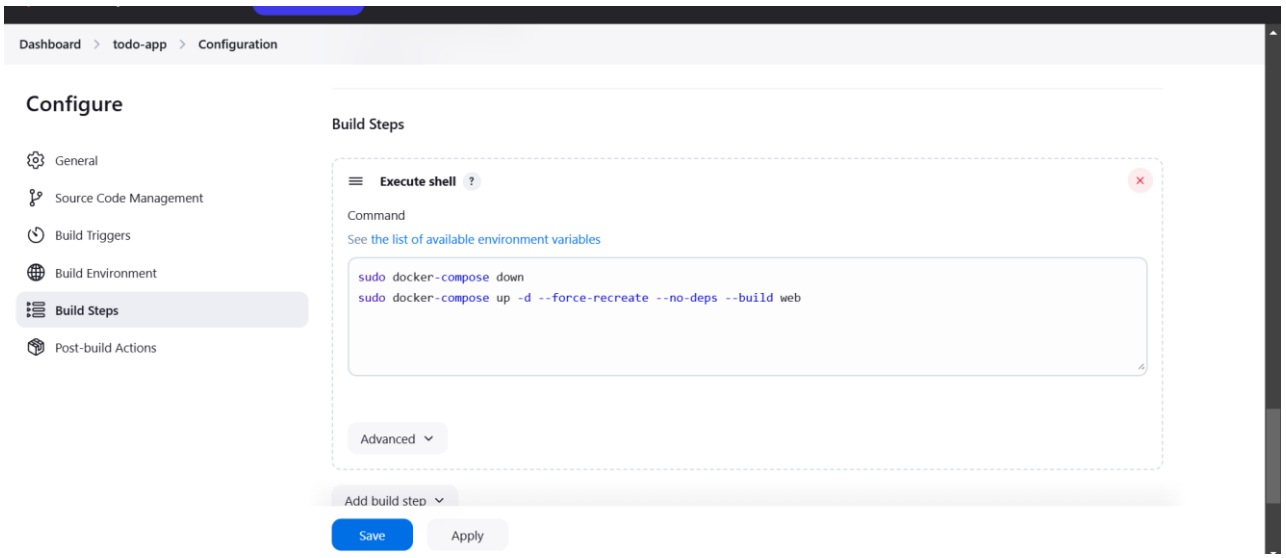


Seventh Step: Automate with Docker Compose

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
```



- **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**