AM

Anand Wadsinge's Blog

Follow

Project on building a CI/CD pipeline for Django App with Jenkins on Docker Container

AW Anand Wadsinge

Sep 25, 2023 • 4 min read

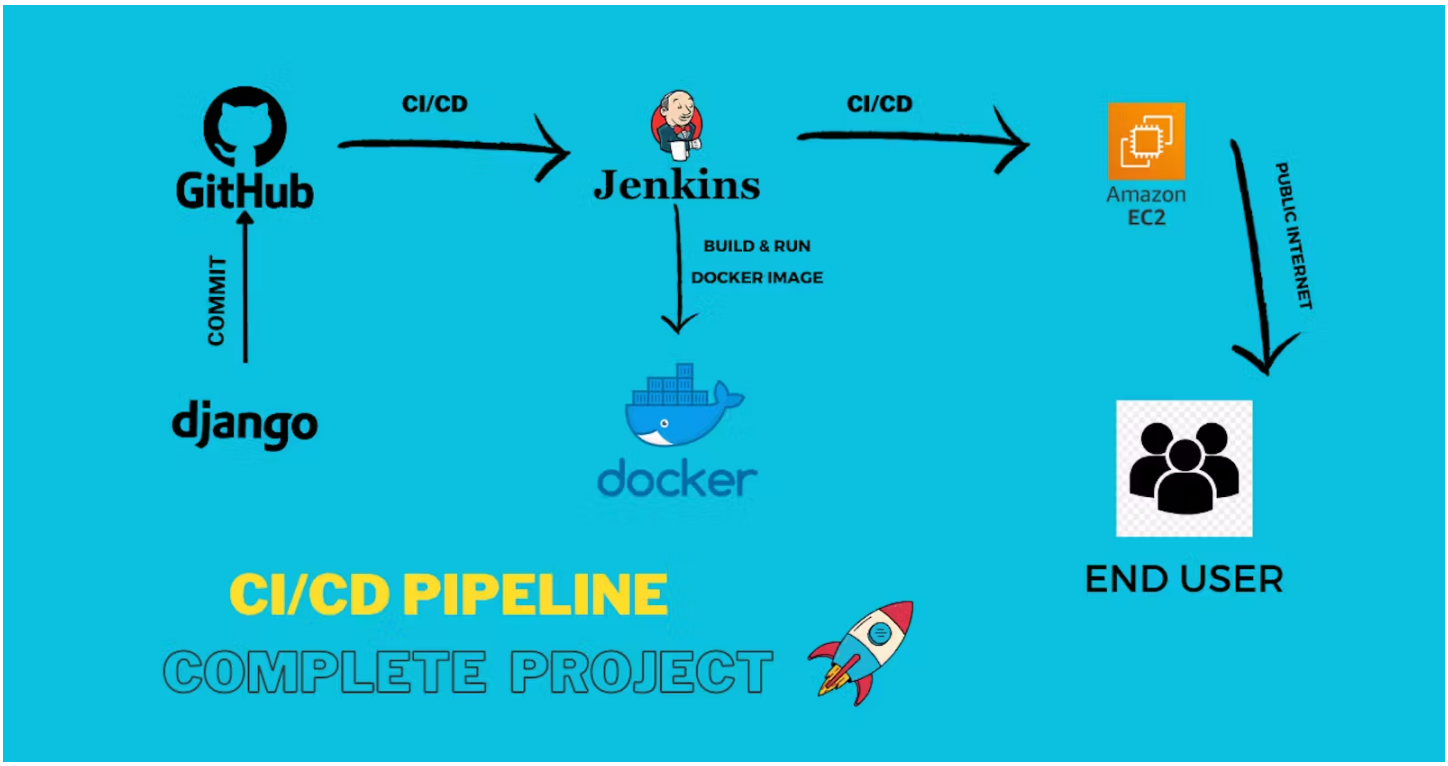


TABLE OF CONTENTS

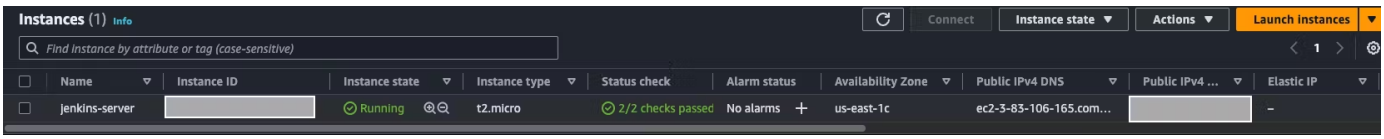
In this blog, we are going to deploy a **Django** Web app on a **Docker Container** built on an **EC2** Instance through the use of **Jenkins**.

Agenda:

- Setup EC2 Instance
- Install Jenkins
- Integrate GitHub Webhook with Jenkins
- Setup Docker Host
- Automate and Build the deployment process
- Test Deployment

Step 1: Setup Jenkins Server on AWS EC2 Instance

- First go to AWS portal and create a EC2 Instance



- Allow following ports from the security group under “Inbound Rule”
Here, 8080 will be used by Jenkins server and 8000 will be used by Django App

Inbound rules

Filter rules

Name	Security group rule ID	Port range	Protocol	Source	Security groups	Description
–	sgr-019440739221ed0bd	80	TCP	0.0.0.0/0	launch-wizard-2	–
–	sgr-034999298d24160fd	443	TCP	0.0.0.0/0	launch-wizard-2	–
–	sgr-0b43368c31cfc5b2d	8000	TCP	0.0.0.0/0	launch-wizard-2	–
–	sgr-00a1f188809e22f10	22	TCP	0.0.0.0/0	launch-wizard-2	–
–	sgr-00ae4569fbfa3cc0b	8080	TCP	0.0.0.0/0	launch-wizard-2	–

- Now connect to the EC2 Instance that you have created using SSH

- Install Jenkins using the following link: https://www.jenkins.io/doc/book/installing/linux/

- As a prerequisite first install Java because Jenkins require Java to run - for installation use below command and then proceed with Jenkins installation:

```
sudo apt install openjdk-17-jre
```

* Or you can use below script as well to run all the installation command at once.

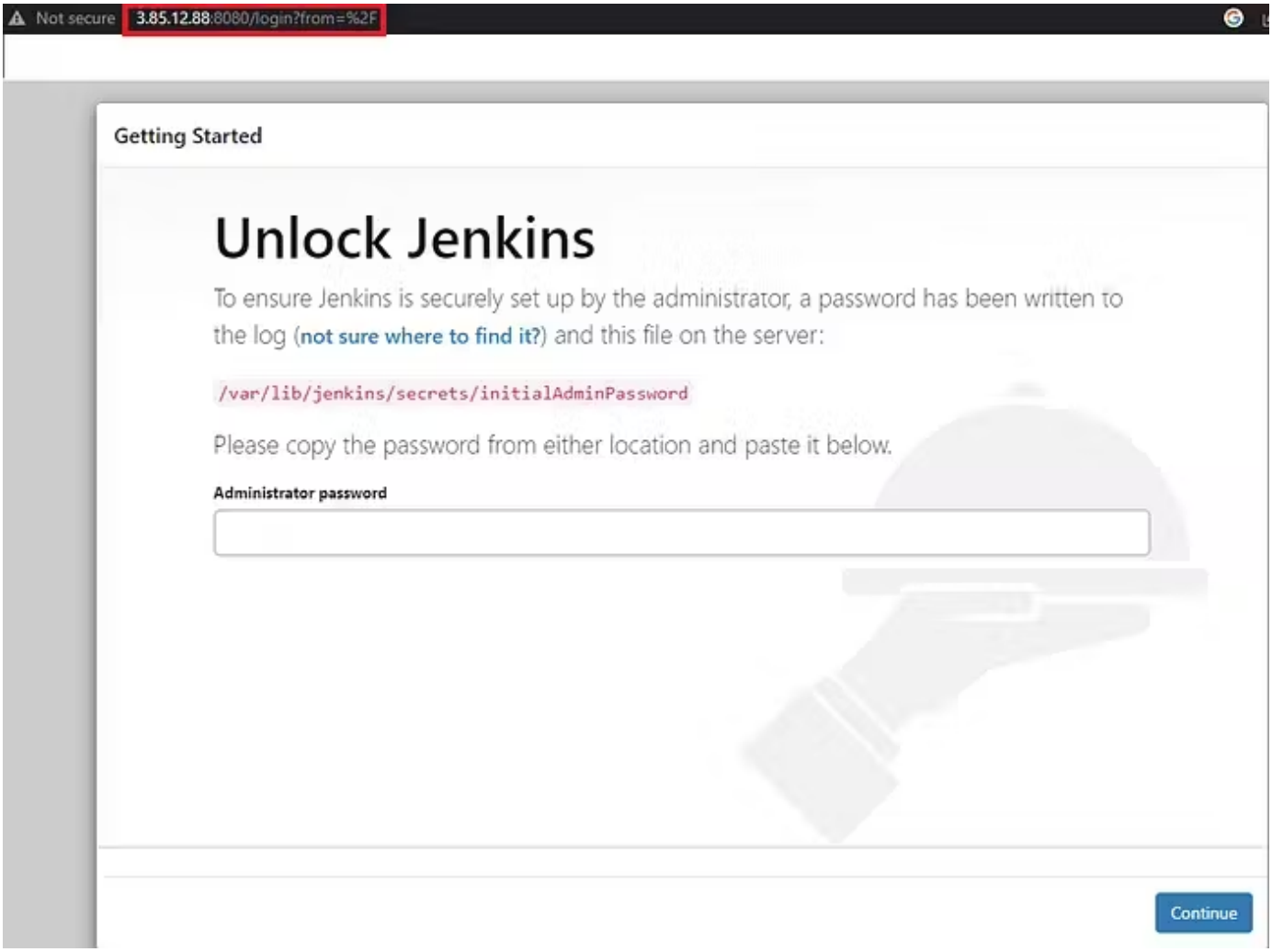
```
#!/bin/bash

sudo apt install openjdk-17-jre && \

curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee \
  /usr/share/keyrings/jenkins-keyring.asc > /dev/null
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt-get update -y && \
sudo apt-get install jenkins
~
```

\> sh installation.sh

- Verify the installed version

```
ubuntu@ip-172-31-91-241:~$ java --version
openjdk 17.0.8.1 2023-08-24
OpenJDK Runtime Environment (build 17.0.8.1+1-Ubuntu-0ubuntu122.04)
OpenJDK 64-Bit Server VM (build 17.0.8.1+1-Ubuntu-0ubuntu122.04, mixed mode, sharing)
ubuntu@ip-172-31-91-241:~$ jenkins --version
2.414.2
ubuntu@ip-172-31-91-241:~$
```

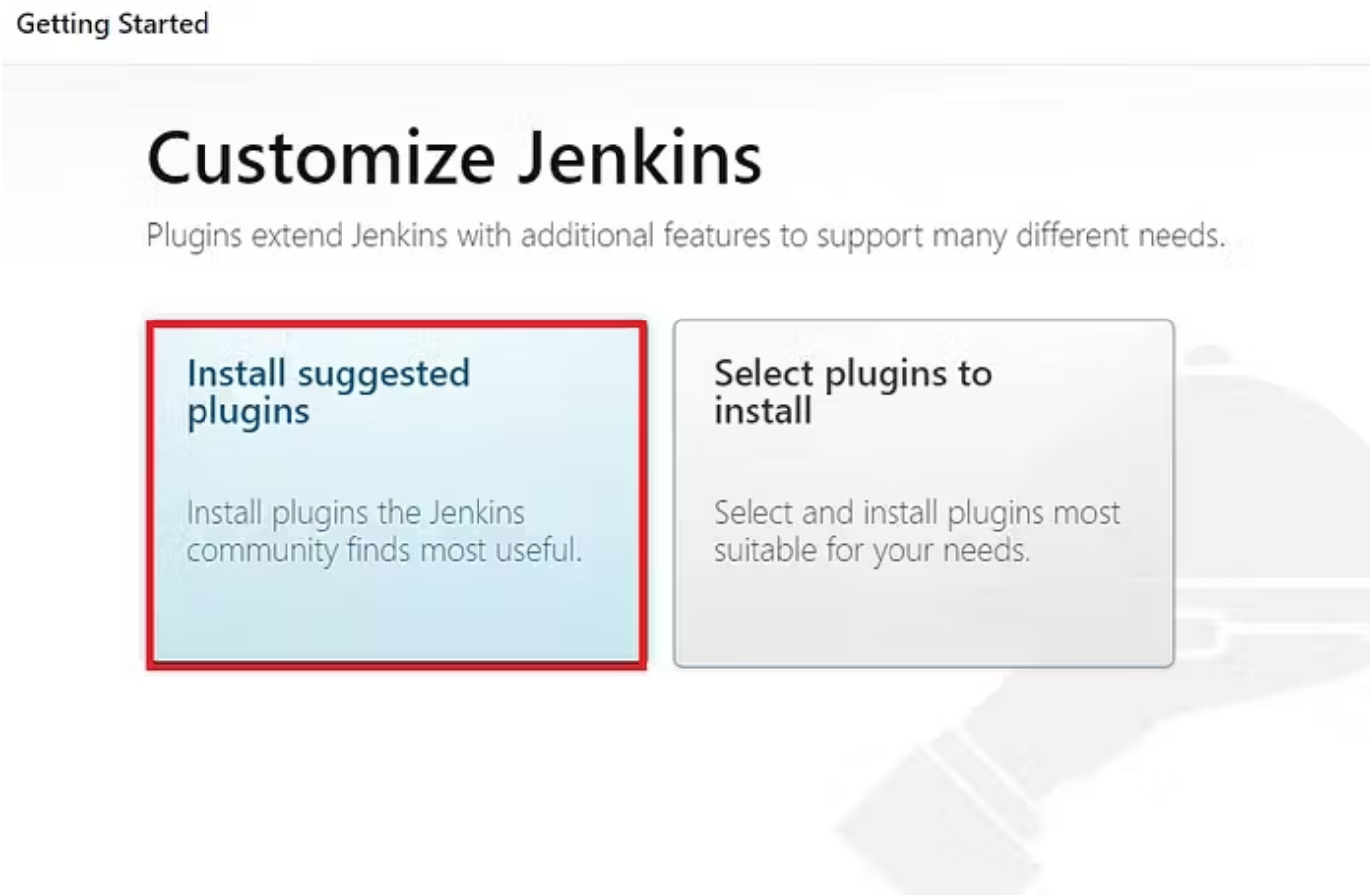
- Once Jenkins is installed try to access the Jenkins server through the browser using <ec2_ip_address>:8080



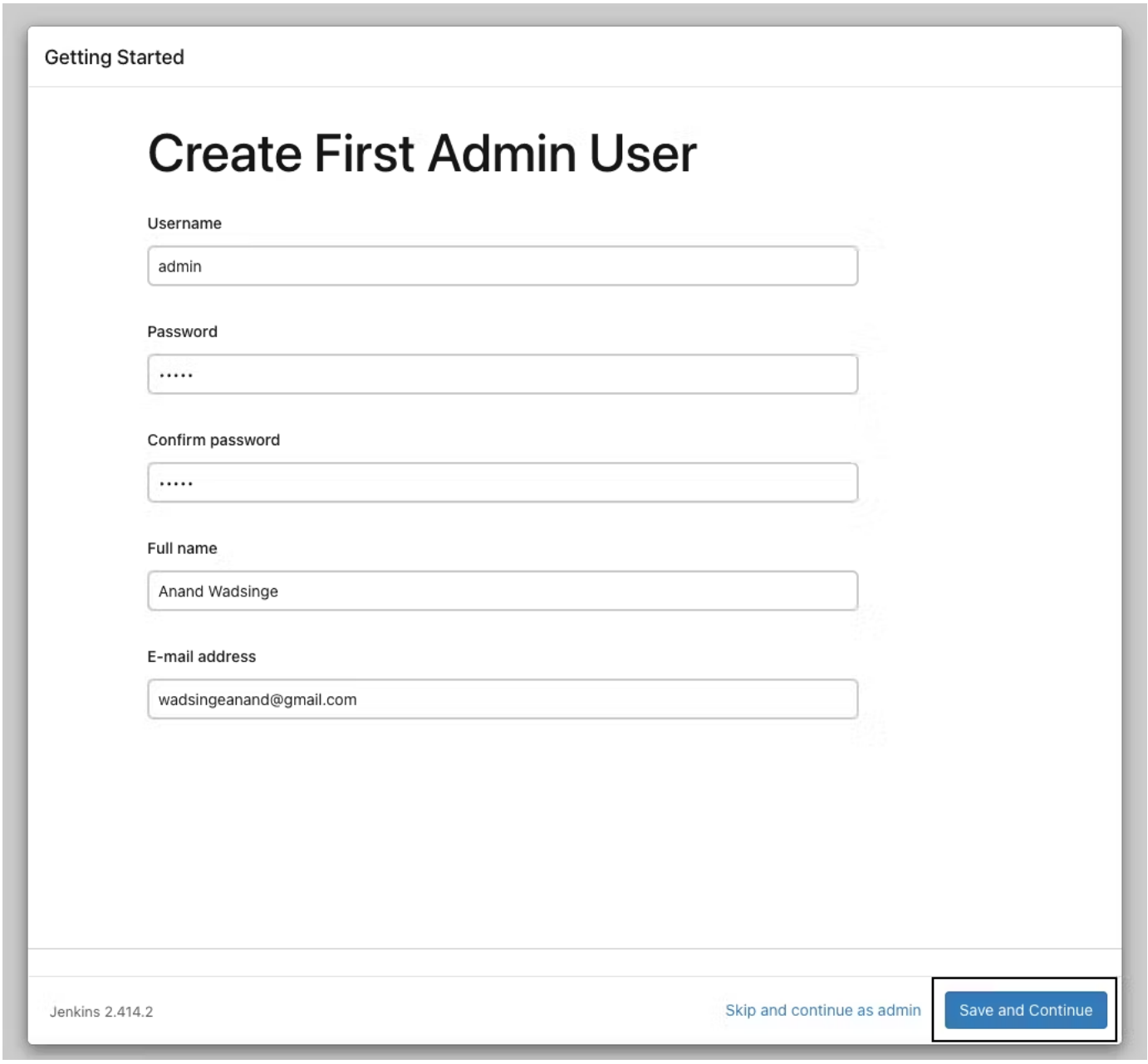
- To unlock Jenkins we need to go to the path /var/lib/jenkins/secrets/initialAdminPassword and fetch the admin password to proceed further

```
[root@ip-172-31-19-12                                    nitialAdminPassword
8c4a40fe0a4e4bea97c67
```

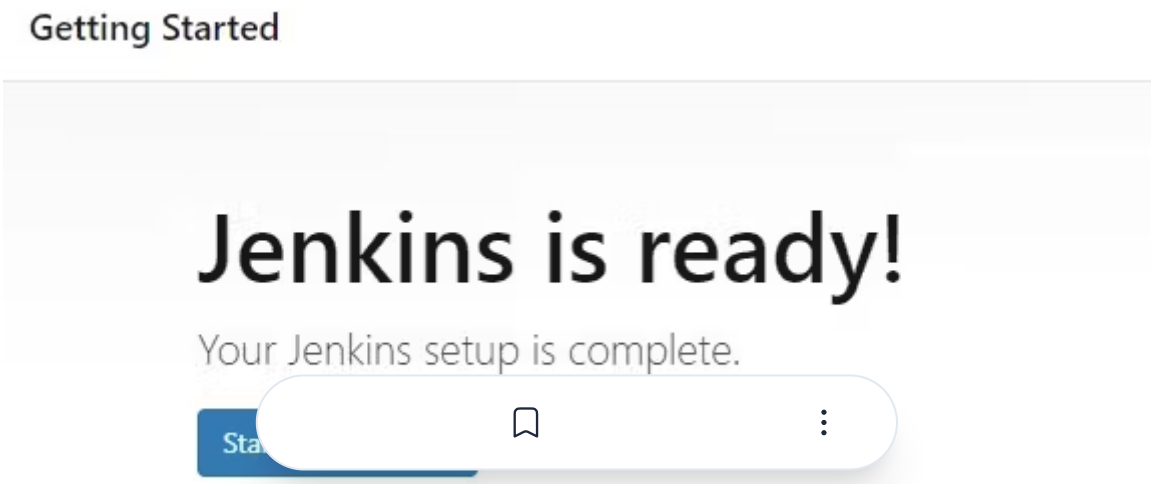
- Now on the Customize Jenkins page, we can go ahead and install the suggested plugins:



- Now we can create our first Admin user, provide all the required data and proceed to save and continue.

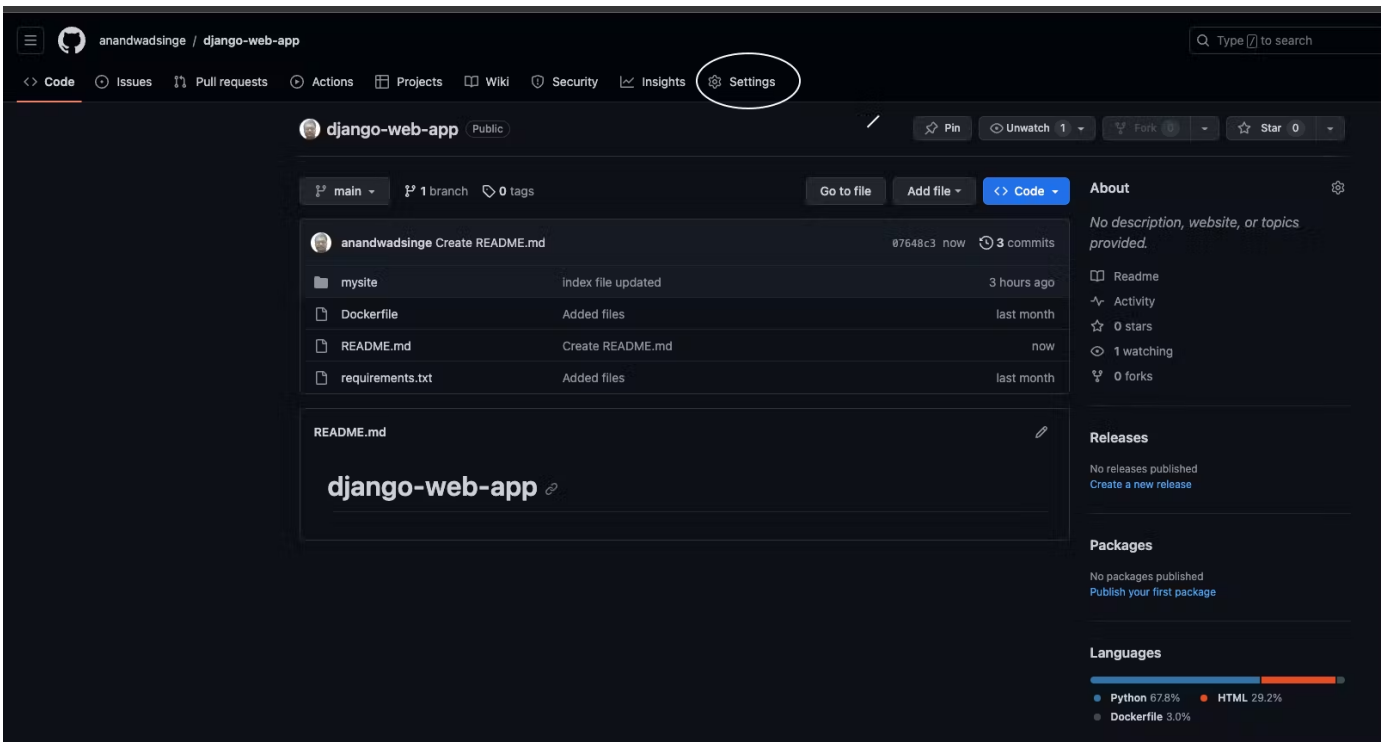


- Now we are ready to use our Jenkins server



Step 2: Integrate GitHub Webhooks with Jenkins

Goto GitHub repository of your Django project and open the settings



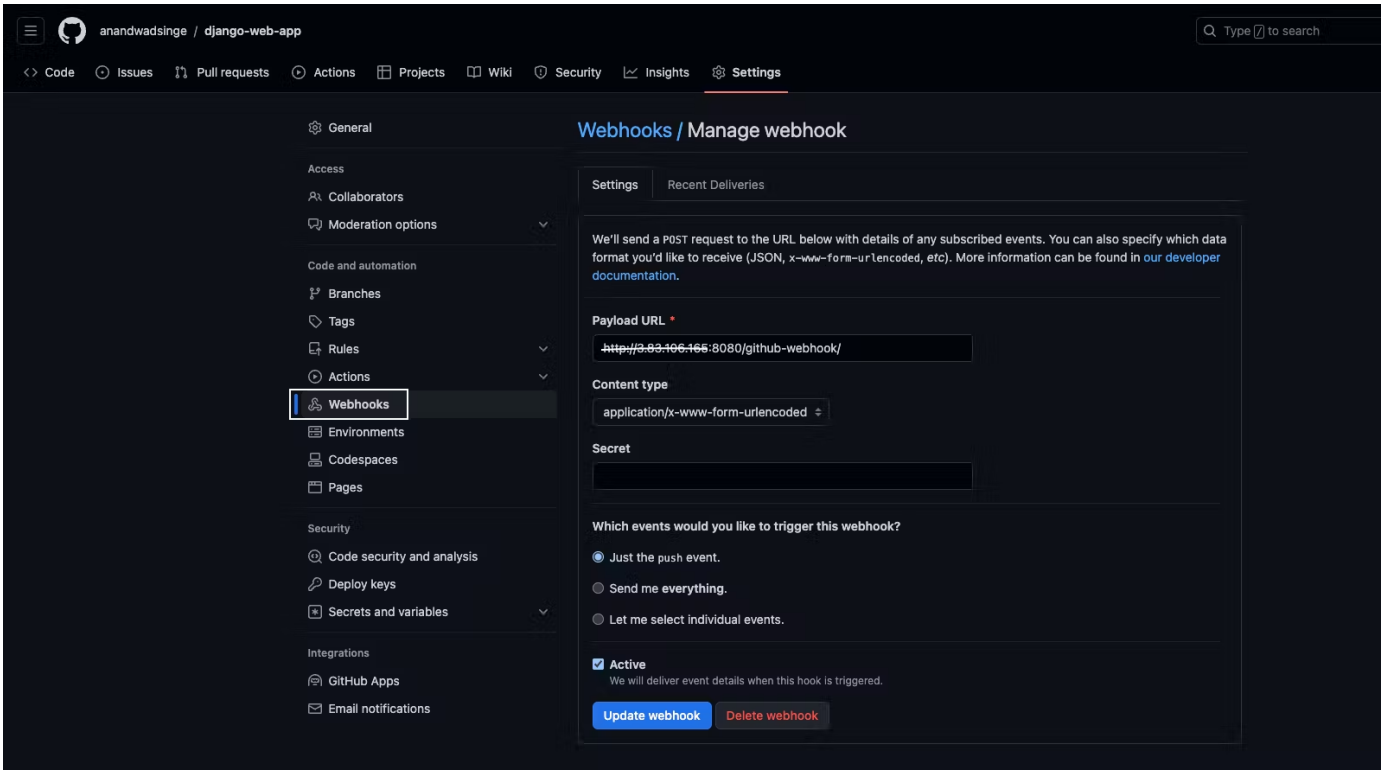
If you don't have Django project you can also fork/clone this repository at your end.

> https://github.com/anandwadsinge/django-web-app.git

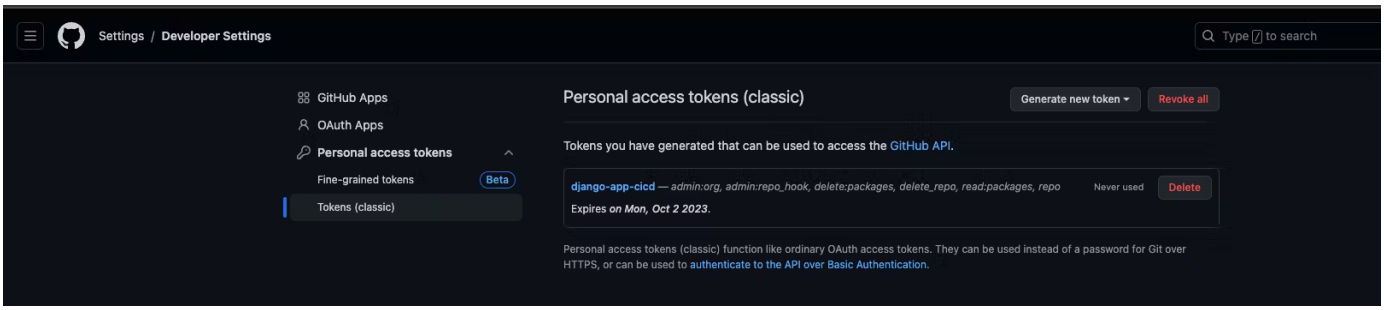
In the **repository settings** select **Webhooks > Add webhook --** in the payload URL section enter

`http://<ip_addr>:8080/github-webhook/` and then click on Add webhook.

Here, we are linking this repository to our Jenkins server.

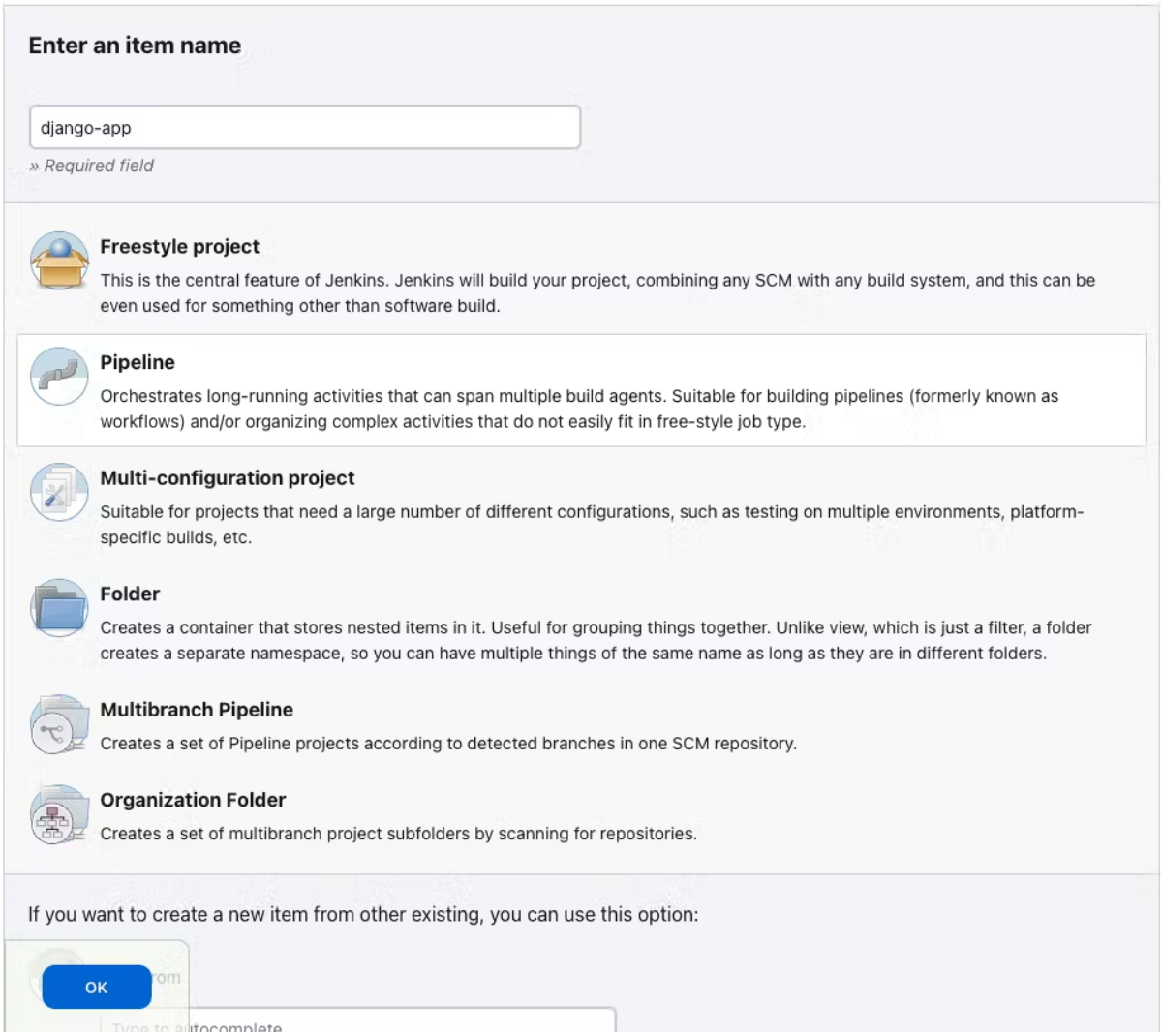


Now, to authenticate with Jenkins server we will need to create a Personal Access Token on Github. So, to create an Token goto **Profile Settings > Developer Settings > Personal access tokens > Tokens (classic) > Generate access tokens (classic)** with the scopes that are required.

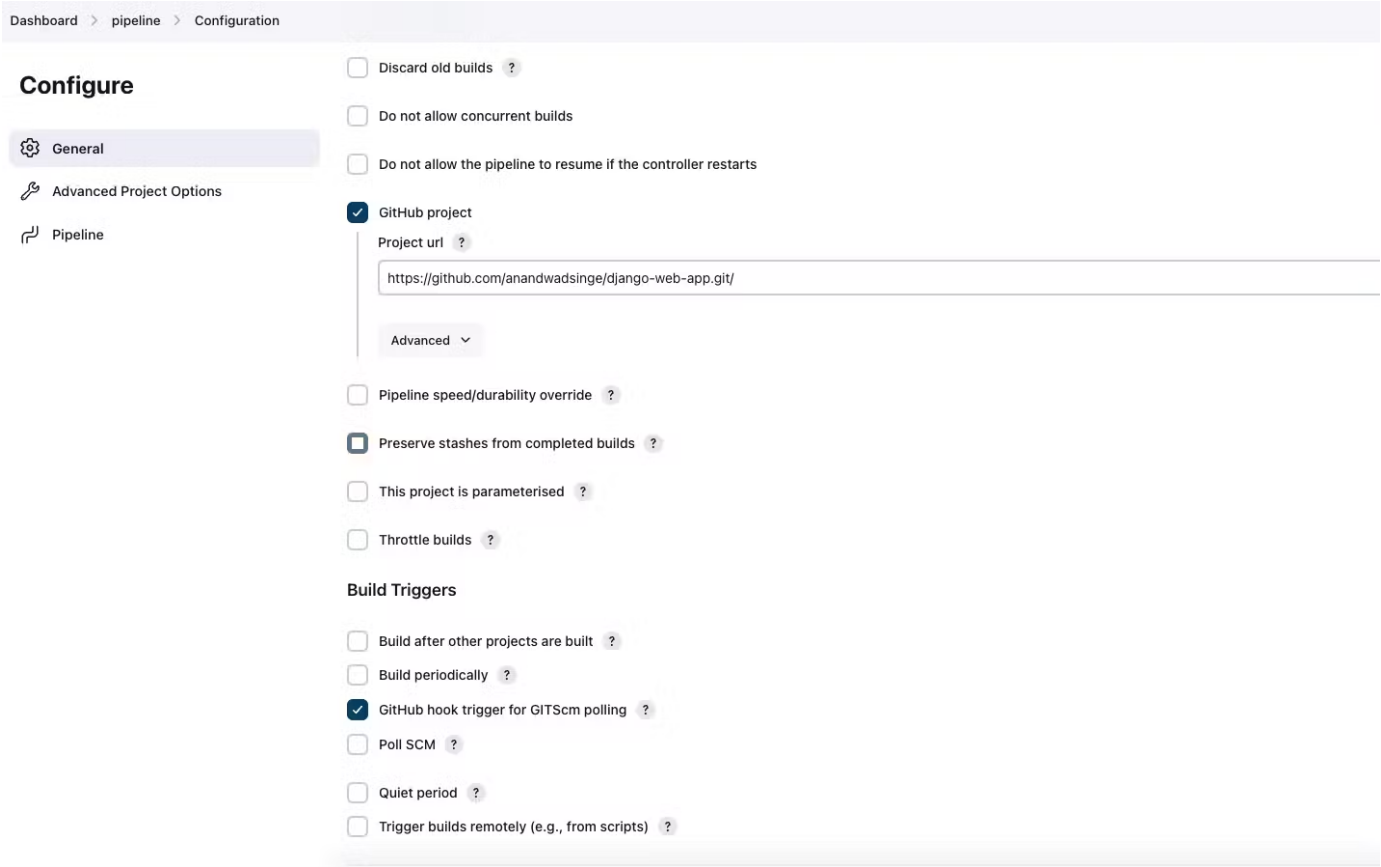


Step 3: Configuration on CI/CD Pipeline

Create a new job and configure your pipeline

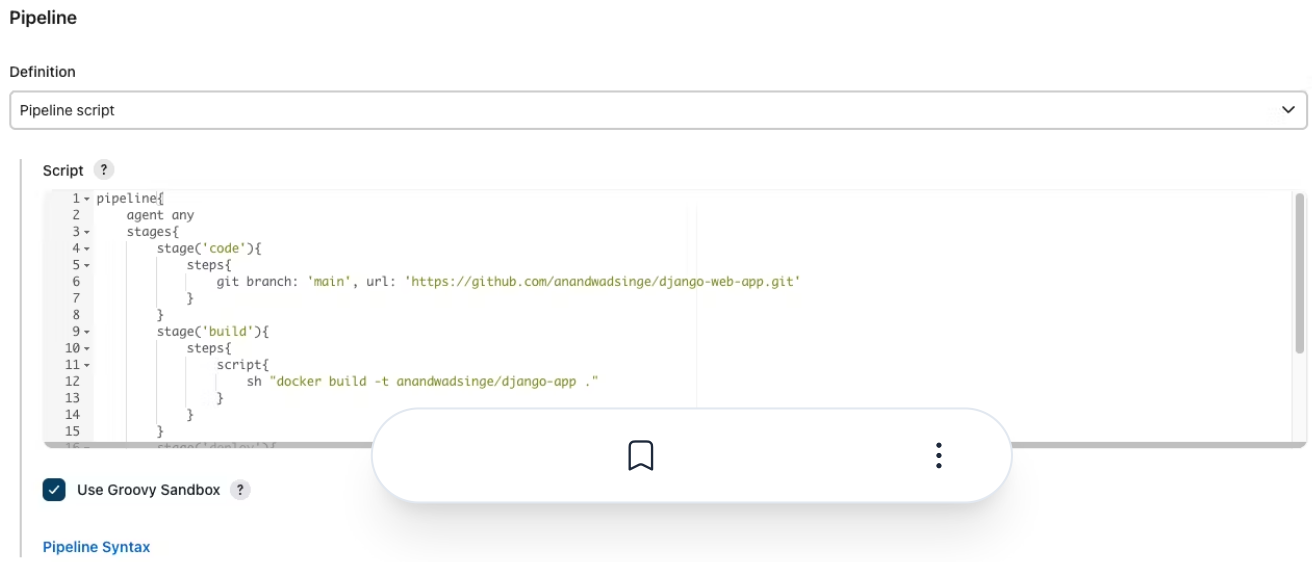


Check **Github project** checkbox and add your project URL. Under **build triggers** select **GitHub hook trigger for GITScm polling >** which means that every time Jenkins receives a PUSH GitHub hook (from the repository you defined in the Source Code Management section) it will trigger the polling



In Pipeline definition, select pipeline script, and enter your groovy code.

```
pipeline{ agent any stages{ stage('code'){ steps{ git branch: 'main', url: ' https://github.com/anandwadsinge/django-web-app.git ' } } stage('build'){ steps{ script{ sh "docker build -t anandwadsinge/django-app ." } } } stage('deploy'){ steps{ script{ sh "docker run -p 8000:8000 -d anandwadsinge/django-app" } } } } }
```



pipeline{ Untitled-1

```
pipeline{
    agent any
    stages{
        stage('code'){
            steps{
                git branch: 'main', url: 'https://github.com/anandwadsinge/django-web-app.git'
            }
        }
        stage('build'){
            steps{
                script{
                    sh "docker build -t anandwadsinge/django-app ."
                }
            }
        }
        stage('deploy'){
            steps{
                script{
                    sh "docker run -p 8000:8000 -d anandwadsinge/django-app"
                }
            }
        }
    }
}
```

Apply and save this job.

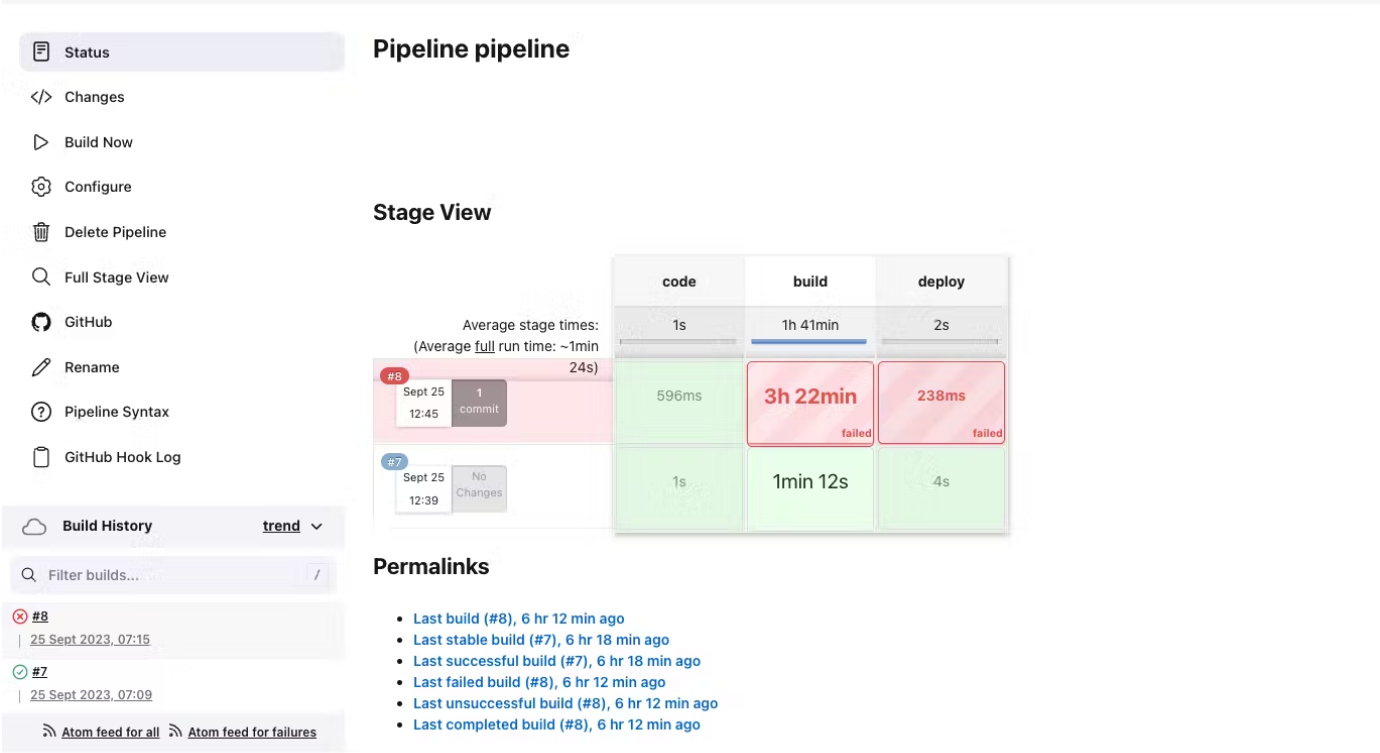
Now, before building this job we need to install docker and user to the docker on the node using the following commands:

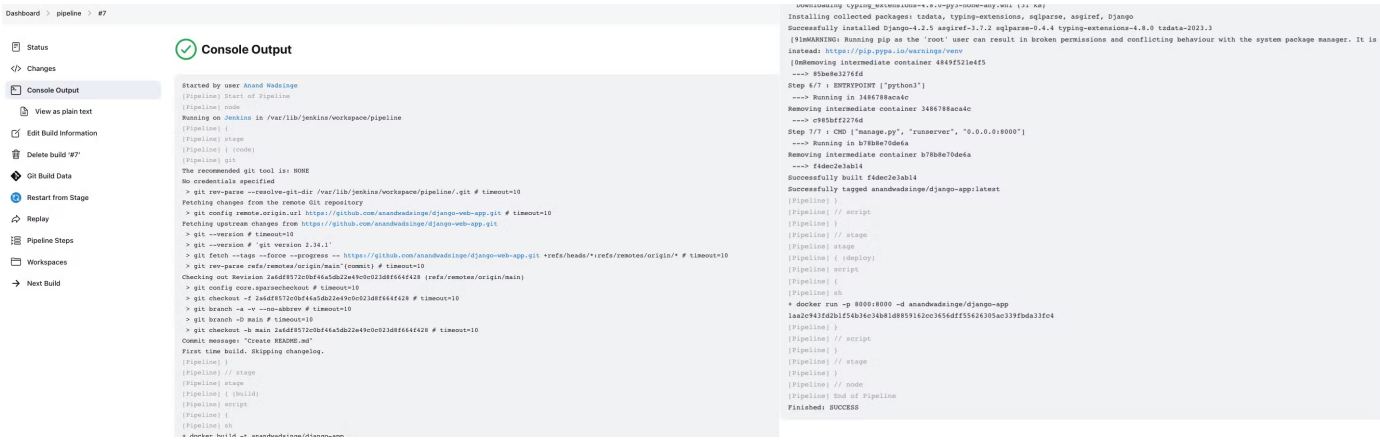
COPY

```
sudo apt-get install docker.io -y
sudo apt-get update -y
sudo usermod -aG docker jenkins
sudo usermod -aG root jenkins
sudo chmod 664 /var/run/docker.sock
sudo reboot
```

Once the reboot of server is done we are ready to build our job pipeline.

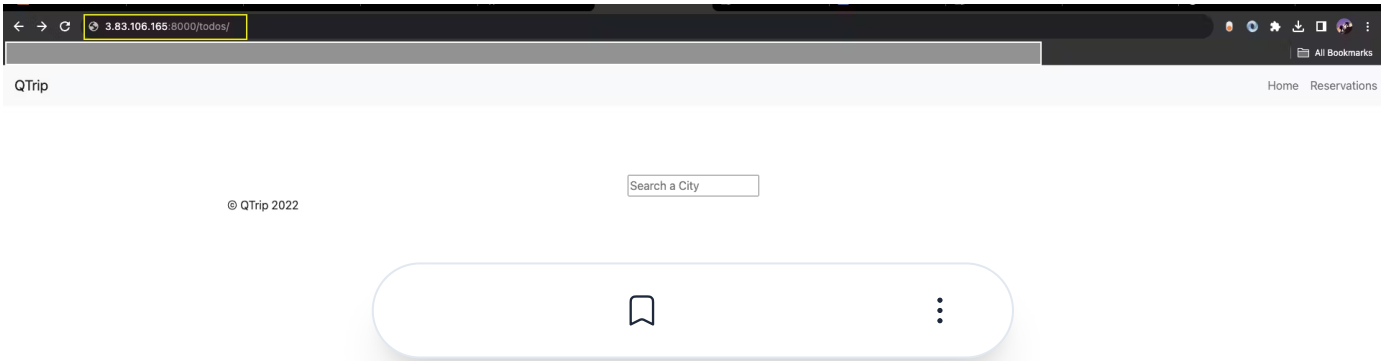
Click on **Build now** to start the build process.





The Django-app was successfully build and deployed.

Finally, we can see the application is running on browser - URL ‘http://<public_ip_address>:8000’



Conclusion: In this blog, we learnt how to install Jenkins, as well as how to integrate git webhooks with Jenkins and configure jobs in the Jenkins pipeline.

If you face any problems or have any questions please let me know in the comments section I will try and answer it.

Thanks for reading. I hope you find this article useful.

Jenkins Docker Django Python GitHub

AW

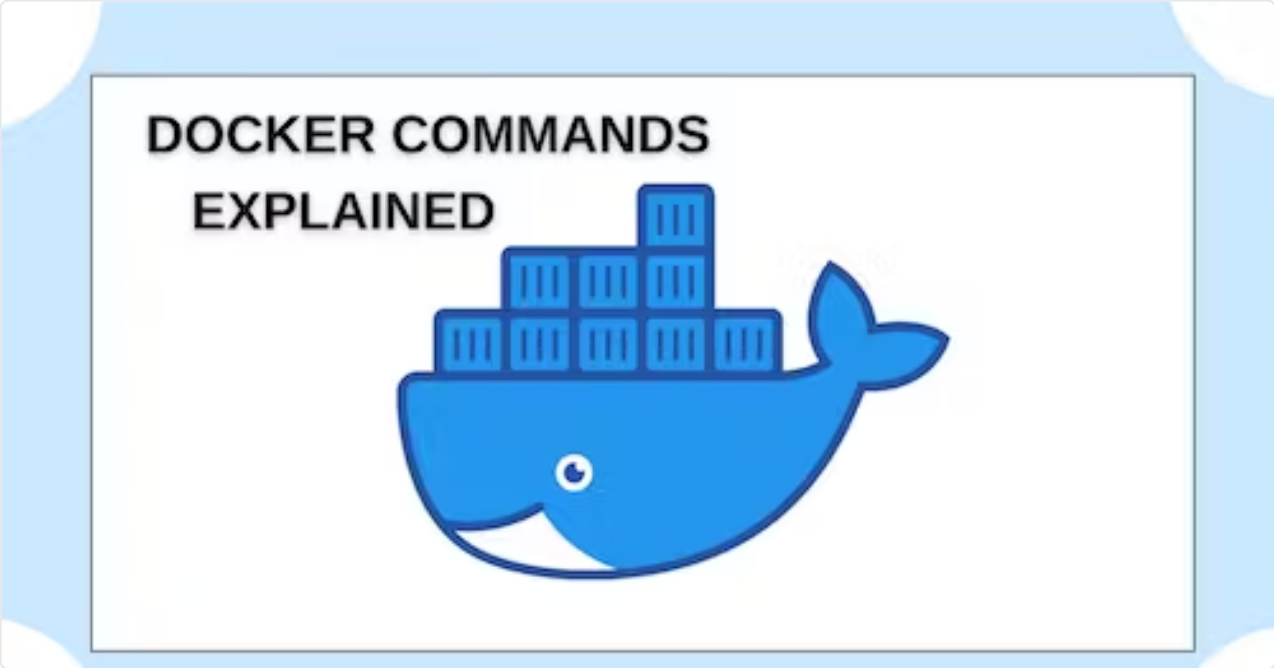
WRITTEN BY

Anand Wadsinge

Follow

I am passionate about tech and love writing about my experience and project which I aim to build using latest tech-stacks.

AW **Anand Wadsinge**



Mastering Docker: Essential Commands for Daily Operations🚀🐳

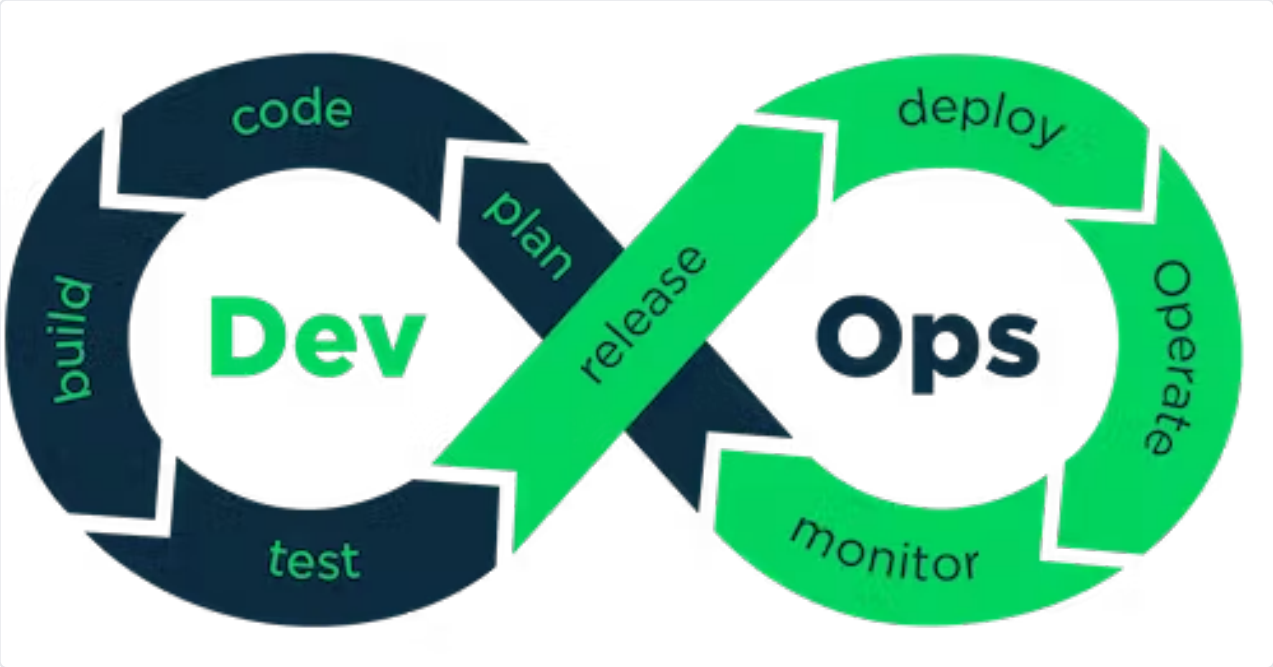
These are some essential Docker commands that will streamline your day-to-day operations and boost y...

AW **Anand Wadsinge**

GIT – src refspe

The error message "src refspec main does not match any" typically occurs when you are trying to push...

AW **Anand Wadsinge**



What is DevOps? How it helps in Automation, Scaling and Infrastructure?

DevOps stands for development (dev) and operations (ops). It is a set of practices that combines sof...