



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

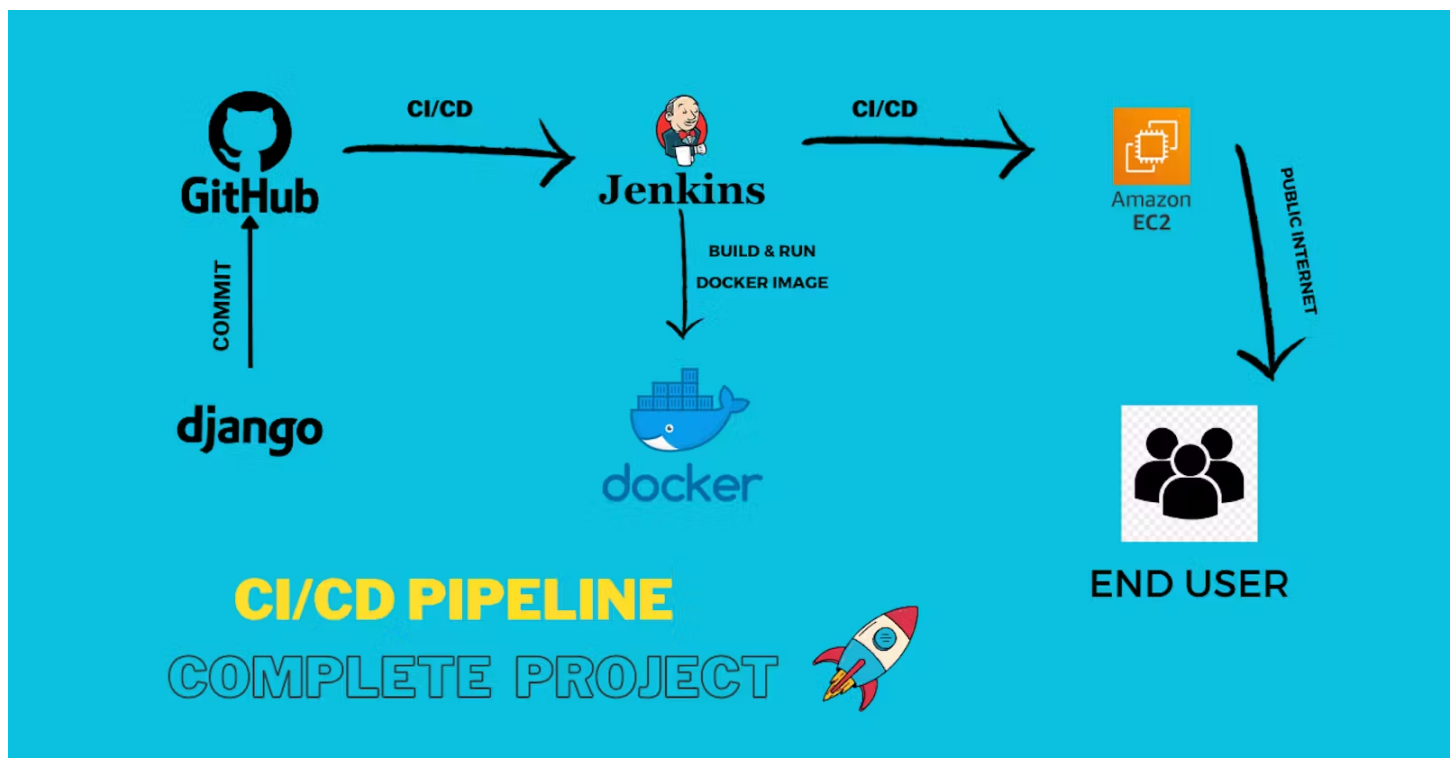


TABLE OF CONTENTS



Step 1: Setup Jenkins Server on AWS EC2 Instance

Step 2: Integrate GitHub Webhooks with Jenkins

Step 3: Configuration on CI/CD Pipeline

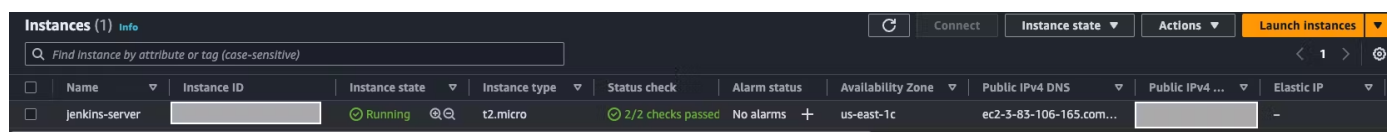
In this blog, we are going to deploy a **Django** Web app on a **Docker Container** built on an **EC2** Instance through the use of **Jenkins**.

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



	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
<input type="checkbox"/>	jenkins-server		Running	t2.micro	2/2 checks passed	No alarms	us-east-1c	ec2-3-83-106-165.com...		-

- 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-0b43368c31fc5b2d	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
```

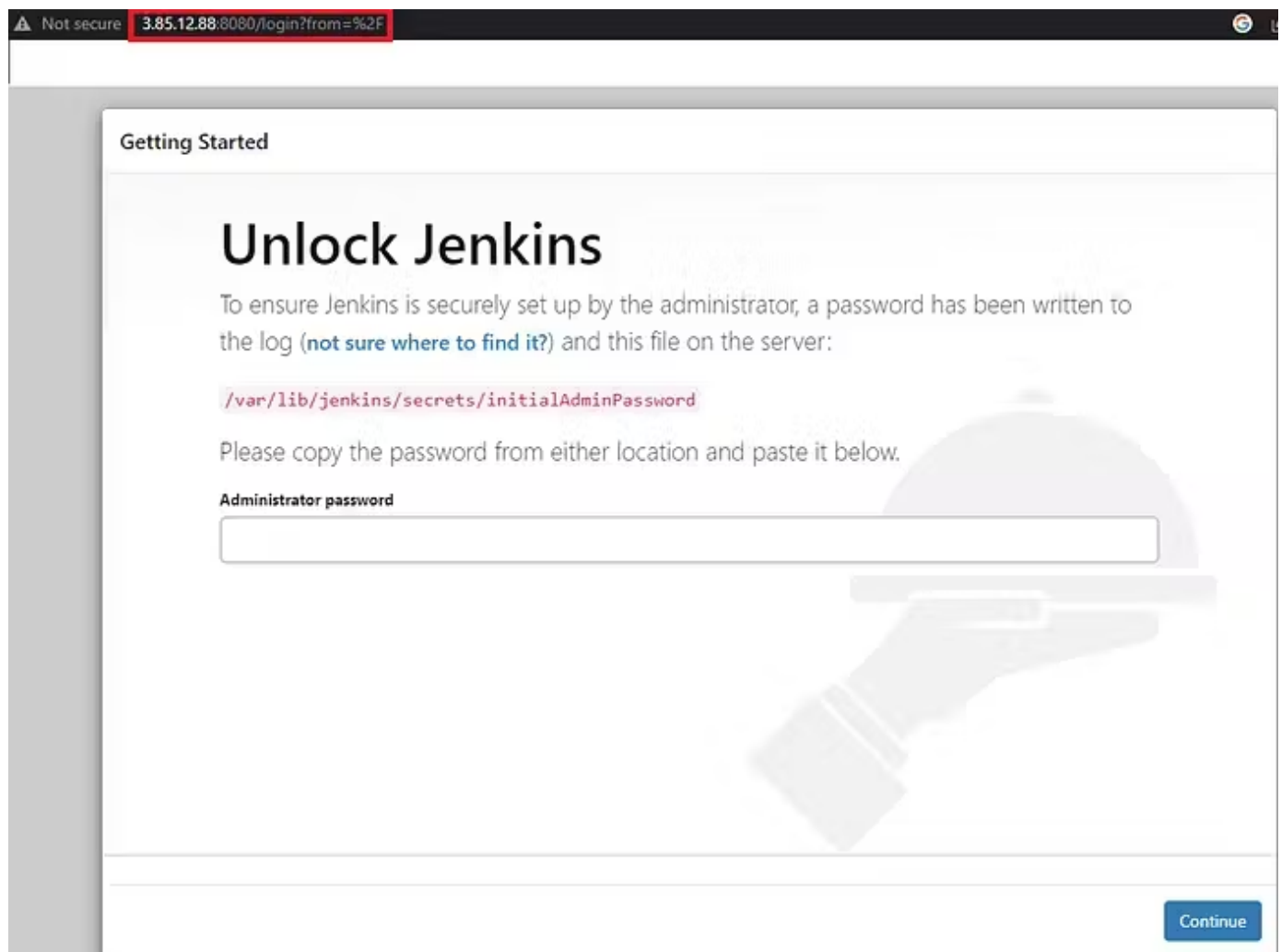
\> sh [installation.sh](#)

- 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 ~]# cat /var/lib/jenkins/secrets/initialAdminPassword
8c4a40fe0a4e4bea97c67...
```

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

Getting Started

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

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



Getting Started

Create First Admin User

Username

Password

Confirm password

Full name

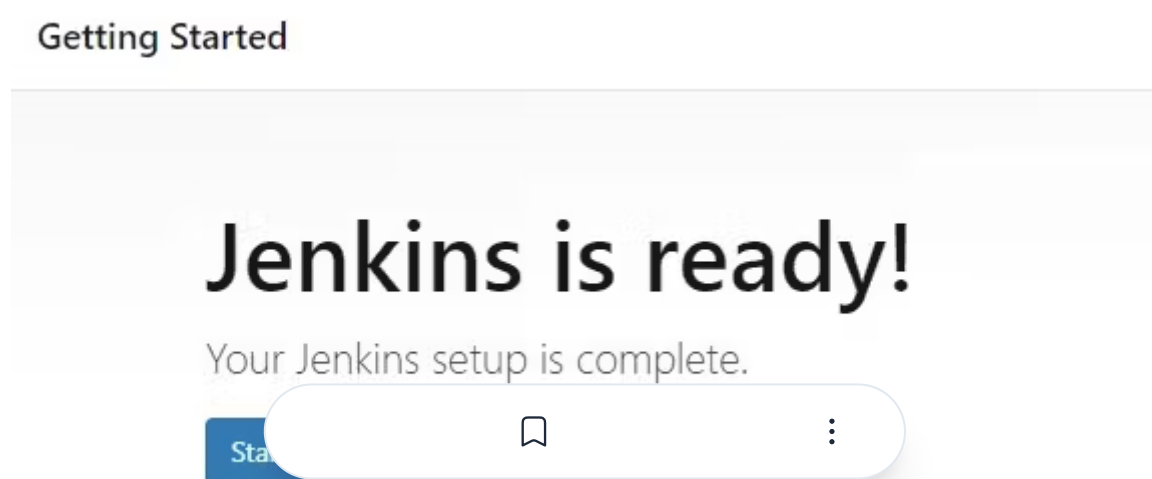
E-mail address

Jenkins 2.414.2

[Skip and continue as admin](#)

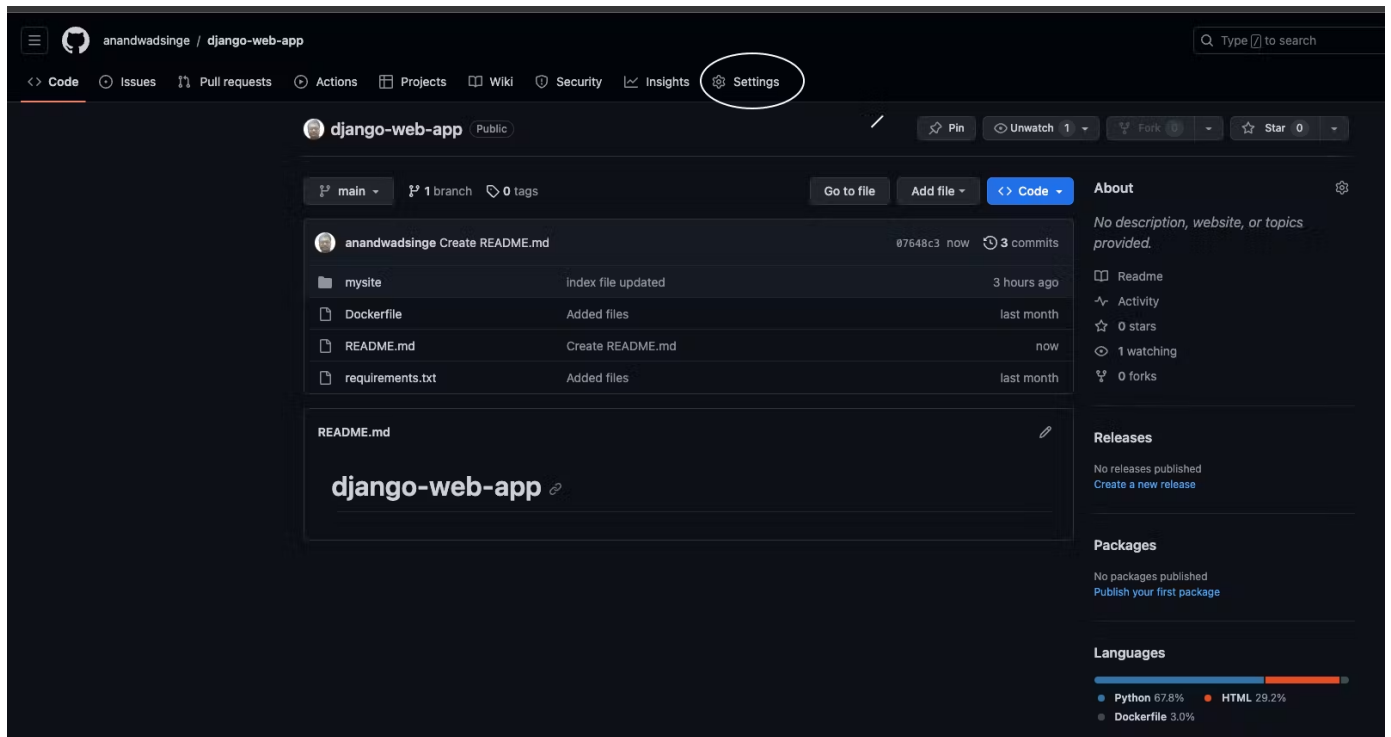
[Save and Continue](#)

- 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.

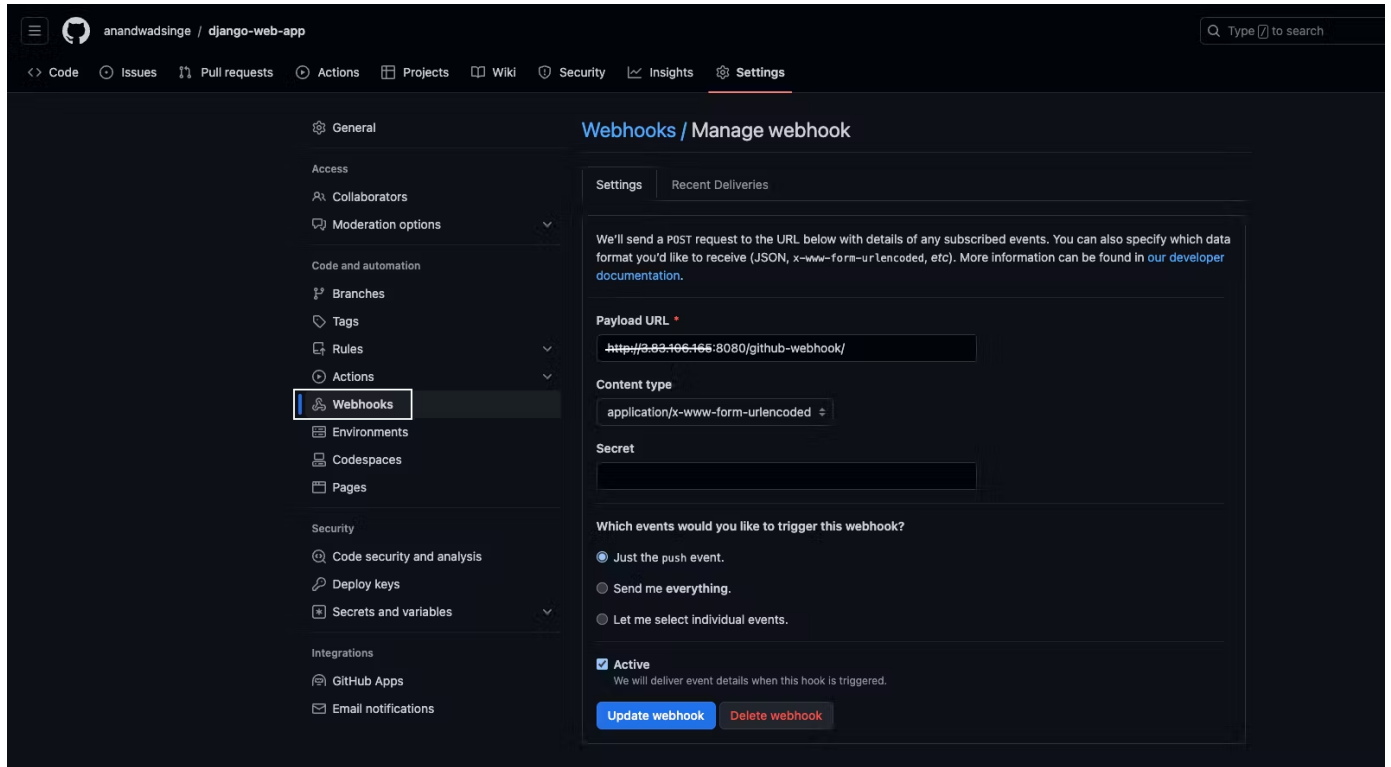
> <https://github.com/anandwadsing/django-web-app.git>

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

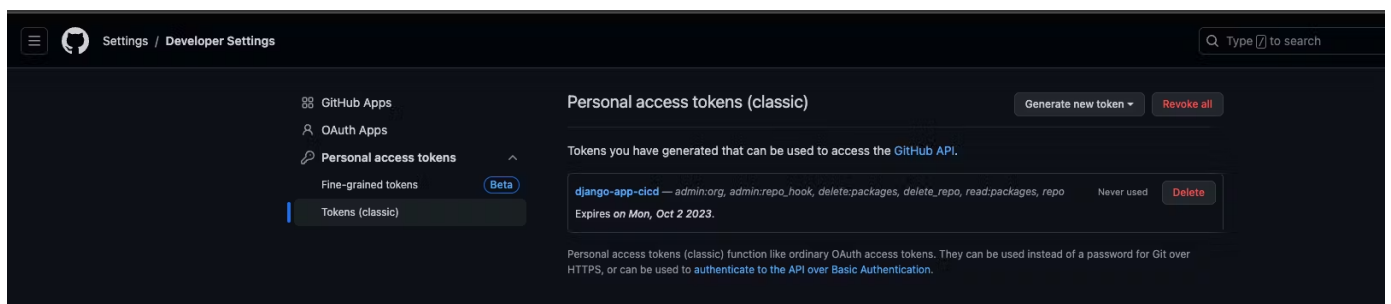
`http://<ip_addr>:8080/github-webhook/` and then click on Add webhook.

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




Enter an item name

django-app


» Required field

**Freestyle project**


This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

**Pipeline**


Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

**Multi-configuration project**


Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

**Folder**

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

**Multibranch Pipeline**

Creates a set of Pipeline projects according to detected branches in one SCM repository.

**Organization Folder**

Creates a set of multibranch project subfolders by scanning for repositories.

If you want to create a new item from other existing, you can use this option:

OK

Cancel

More options

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



Dashboard > pipeline > Configuration

Configure

- General
- Advanced Project Options
- Pipeline

☐ Discard old builds ?
 ☐ Do not allow concurrent builds
 ☐ Do not allow the pipeline to resume if the controller restarts
 ☒ GitHub project

Project url ?

☐ Pipeline speed/durability override ?
 ☒ Preserve stashes from completed builds ?
 ☐ This project is parameterised ?
 ☐ Throttle builds ?

Build Triggers

☐ Build after other projects are built ?
 ☐ Build periodically ?
 ☒ GitHub hook trigger for GITScm polling ?
 ☐ Poll SCM ?
 ☐ Quiet period ?
 ☐ Trigger builds remotely (e.g., from scripts) ?

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/anandwadsing/django-web-app.git ' } } stage('build'){
steps{ script{ sh "docker build -t anandwadsing/django-app ." } } }
stage('deploy'){ steps{ script{ sh "docker run -p 8000:8000 -d
anandwadsing/django-app" } } } } }
  
```

Pipeline

Definition

Pipeline script

Script ?

```

1 pipeline{
2   agent any
3   stages{
4     stage('code'){
5       steps{
6         git branch: 'main', url: 'https://github.com/anandwadsing/django-web-app.git'
7       }
8     }
9     stage('build'){
10      steps{
11        script{
12          sh "docker build -t anandwadsing/django-app ."
13        }
14      }
15    }
  
```

☒ Use Groovy Sandbox ?

[Pipeline Syntax](#)

```
pipeline( Untitled-1 ●
1 pipeline{
2   agent any
3   stages[
4     stage('code'){
5       steps{
6         git branch: 'main', url: 'https://github.com/anandwadsinge/django-web-app.git'
7       }
8     }
9     stage('build'){
10      steps{
11        script{
12          sh "docker build -t anandwadsinge/django-app ."
13        }
14      }
15    }
16    stage('deploy'){
17      steps{
18        script{
19          sh "docker run -p 8000:8000 -d anandwadsinge/django-app"
20        }
21      }
22    }
23  ]
24 }
```

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.



Status

Changes

Build Now

Configure

Delete Pipeline

Full Stage View

GitHub

Rename

Pipeline Syntax

GitHub Hook Log

Pipeline pipeline

Stage View

Average stage times:
(Average full run time: ~1min)

	code	build	deploy
#8 Sept 25 12:45 1 commit 24s	596ms	3h 22min failed	238ms failed
#7 Sept 25 12:39 No Changes	1s	1min 12s	4s

Permalinks

- Last build (#8), 6 hr 12 min ago
- Last stable build (#7), 6 hr 18 min ago
- Last successful build (#7), 6 hr 18 min ago
- Last failed build (#8), 6 hr 12 min ago
- Last unsuccessful build (#8), 6 hr 12 min ago
- Last completed build (#8), 6 hr 12 min ago

Build History

trend

Filter builds...

#8

25 Sept 2023, 07:15

#7

25 Sept 2023, 07:09

Atom feed for all

Atom feed for failures

Status

Changes

Console Output

View as plain text

Edit Build Information

Delete build 'W'

Git Build Data

Restart from Stage

Replay

Pipeline Steps

Workspaces

Next Build

Console Output

Started by user Anand Wadsinge

(Pipeline) Start of Pipeline

(Pipeline) node

Running on Jenkins in /var/lib/jenkins/workspace/pipeline

(Pipeline) [

(Pipeline) stage

(Pipeline) { code

(Pipeline) git

The recommended git tool is: NONE

No credentials specified

> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/pipeline/.git # timeout=10

> git fetch --tags --force --progress -- https://github.com/anandwadsinge/django-web-app.git # timeout=10

> git config remote.origin.url https://github.com/anandwadsinge/django-web-app.git # timeout=10

Fetching upstream changes from https://github.com/anandwadsinge/django-web-app.git

> git --version # timeout=10

> git --version # 'git version 2.34.1'

> git fetch --tags --force --progress -- https://github.com/anandwadsinge/django-web-app.git --ref=refs/heads/*:refs/remotes/origin/* # timeout=10

> git rev-parse refs/remotes/origin/main (commits) # timeout=10

Checking out Revision 2a6d93720b64a5d2e490c023d8f664f28 (refs/remotes/origin/main)

> git config core.sparsecheckout # timeout=10

> git checkout -f 2a6d93720b64a5d2e490c023d8f664f28 # timeout=10

> git branch -a -v --no-abbrev # timeout=10

> git branch -D main # timeout=10

> git checkout -b main 2a6d93720b64a5d2e490c023d8f664f28 # timeout=10

Commit message: "Create README.md"

First time build. Skipping changelog.

(Pipeline) [

(Pipeline) // stage

(Pipeline) stage

(Pipeline) { build

(Pipeline) script

(Pipeline) sh

docker build -t anandwadsinge/django-app

Successfully built f4dec2e3ab14

Successfully tagged anandwadsinge/django-app:latest

(Pipeline) [

(Pipeline) // script

(Pipeline) script

(Pipeline) // stage

(Pipeline) stage

(Pipeline) { deploy

(Pipeline) script

(Pipeline) [

(Pipeline) sh

+ docker run -p 8000:8000 -d anandwadsinge/django-app

la2c9af62db1f54b36c36d8f5951d2cc3656d4f55626395ac19f2da31f64

(Pipeline) [

(Pipeline) // script

(Pipeline) [

(Pipeline) // stage

(Pipeline) [

(Pipeline) // node

(Pipeline) End of Pipeline

Finished: SUCCESS

The Django-app was successfully build and deployed.

Finally, we can see the application is running on browser - URL

'http://<public_ip_address>:8000'

3.83.106.165:8000/todos/

QTrip

Home Reservations

© QTrip 2022

Search a City

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.

Subscribe to my newsletter

Read articles from **Anand Wadsinge's Blog** directly inside your inbox. Subscribe to the newsletter, and don't miss out.

SUBSCRIBE

Jenkins

Docker

Django

Python

GitHub



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.

**MORE ARTICLES**

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

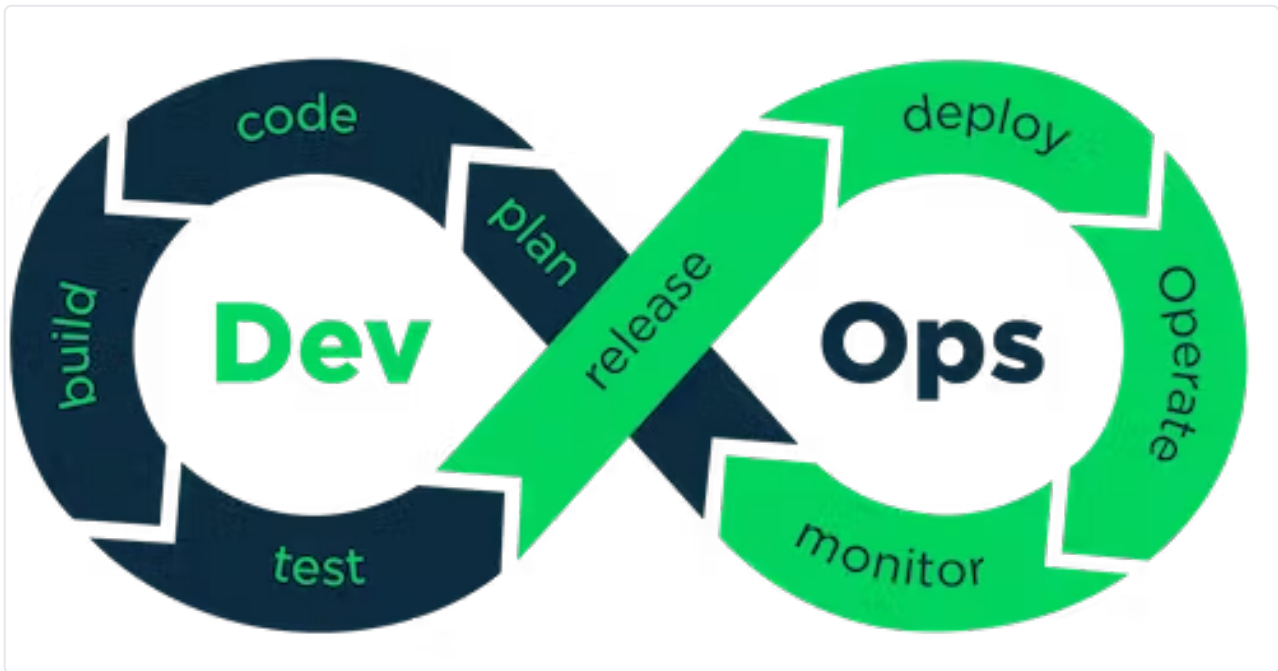


GIT – src refspec



The error message "src refs spec 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...

©2023 Anand Wadsinge's Blog

[Archive](#) · [Privacy policy](#) · [Terms](#)

 Publish with Hashnode

Power



aders