https://www.linkedin.com/in/mohasin-mudassar/

DevOps Project – 1

Automated CI/CD Pipeline for Django Application with Jenkins and GitHub Webhooks

mohasinmudassar16@gmail.com

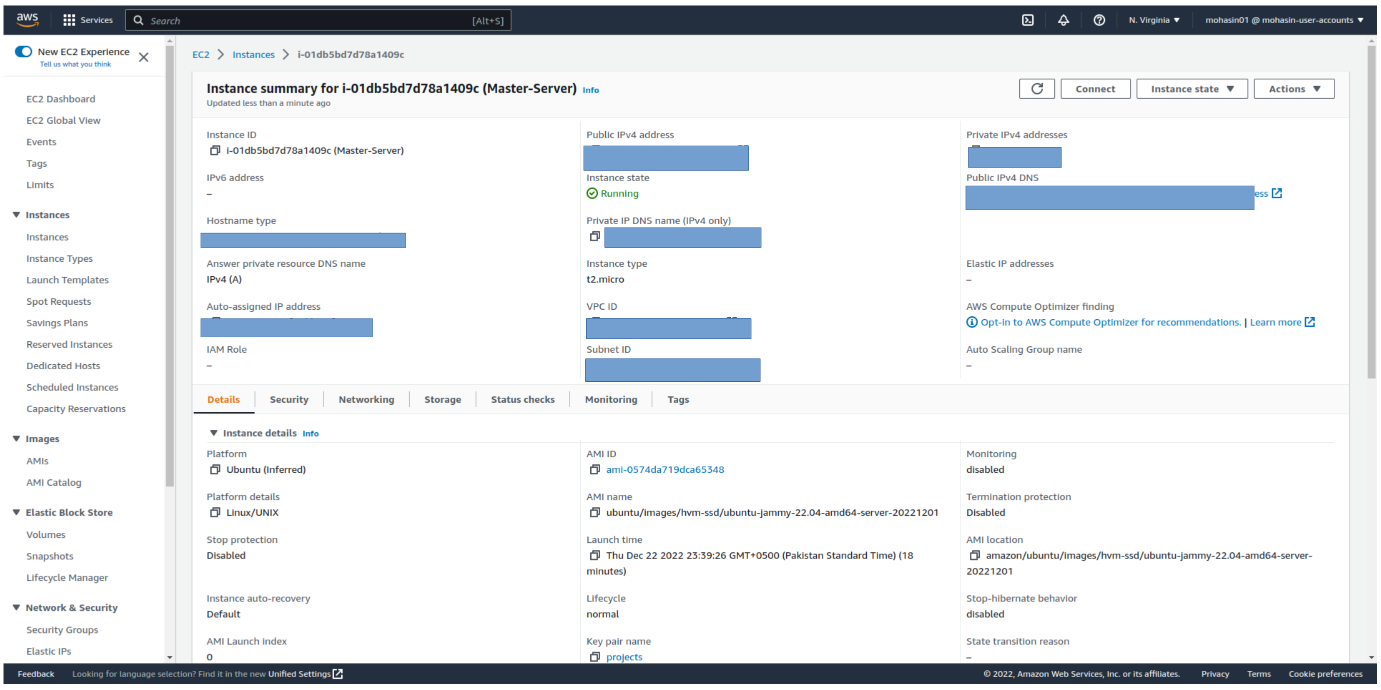
Jenkins CI/CD pipeline with GitHub webhook integration for Deploying Docker application on EC2 instances using the declarative pipeline.

Steps:

1. First of all, go to AWS portal, and create a new instance. As,

- Name: **Master-Server**
- AMI: **ubuntu**.
- Instance type: **t2.micro (free tier)**.
- Key pair login: Create > projects.pem.

(**Download the .pem file**.)
Click on **Launch Instance**.



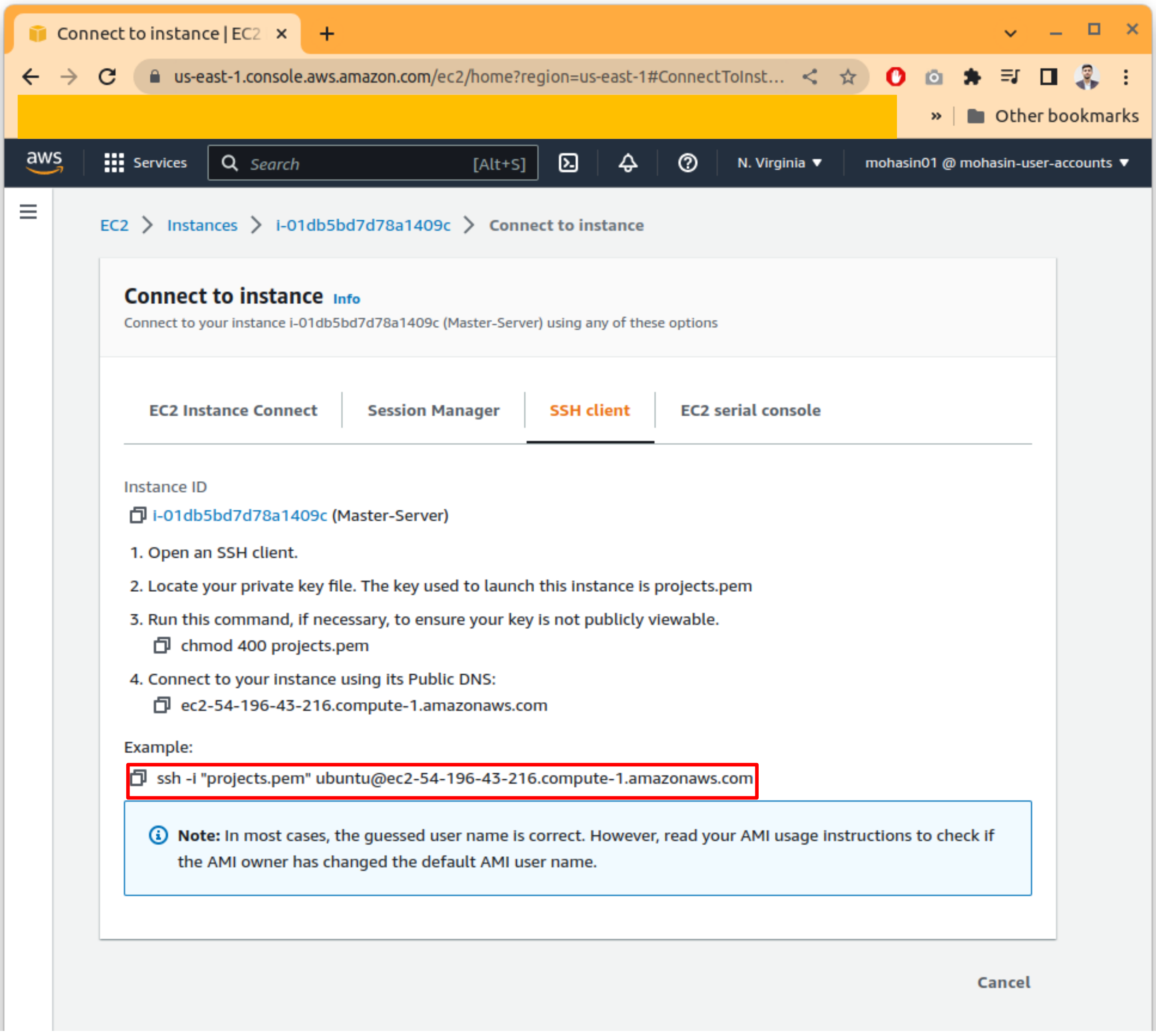
Instance summary for i-01db5bd7d78a1409c (Master-Server) Info
Updated less than a minute ago

Instance ID: i-01db5bd7d78a1409c (Master-Server)
Public IPv4 address: [redacted]
Private IPv4 addresses: [redacted]
IPv6 address: –
Instance state: Running
Public IPv4 DNS: [redacted]ess
Hostname type: [redacted]
Private IP DNS name (IPv4 only): [redacted]
Answer private resource DNS name: IPv4 (A)
Instance type: t2.micro
Elastic IP addresses: –
Auto-assigned IP address: [redacted]
VPC ID: [redacted]
AWS Compute Optimizer finding: Opt-in to AWS Compute Optimizer for recommendations. | Learn more
IAM Role: –
Subnet ID: [redacted]
Auto Scaling Group name: –

▼ Instance details Info

Platform: Ubuntu (Inferred)
AMI ID: ami-0574da719dca65348
Monitoring: disabled
Platform details: Linux/UNIX
AMI name: ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-server-20221201
Termination protection: Disabled
Stop protection: Disabled
Launch time: Thu Dec 22 2022 23:39:26 GMT+0500 (Pakistan Standard Time) (18 minutes)
AMI location: amazon/ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-server-20221201
Instance auto-recovery: Default
Lifecycle: normal
Stop-hibernate behavior: disabled
AMI Launch index: 0
Key pair name: projects
State transition reason: –

2. Now, connect to the EC2 instance that you have created. Copy the SSH from server:


Connect to instance | EC2
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#ConnectToInst...
Other bookmarks
aws
Services
Search
[Alt+S]
N. Virginia
mohasin01 @ mohasin-user-accounts
EC2 > Instances > i-01db5bd7d78a1409c > Connect to instance
EC2 Instance Connect
Session Manager
SSH client
EC2 serial console
Cancel

Connect to instance Info

Connect to your instance i-01db5bd7d78a1409c (Master-Server) using any of these options

Instance ID

i-01db5bd7d78a1409c (Master-Server)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is projects.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.

```
chmod 400 projects.pem
```

4. Connect to your instance using its Public DNS:

```
ec2-54-196-43-216.compute-1.amazonaws.com
```

Example:

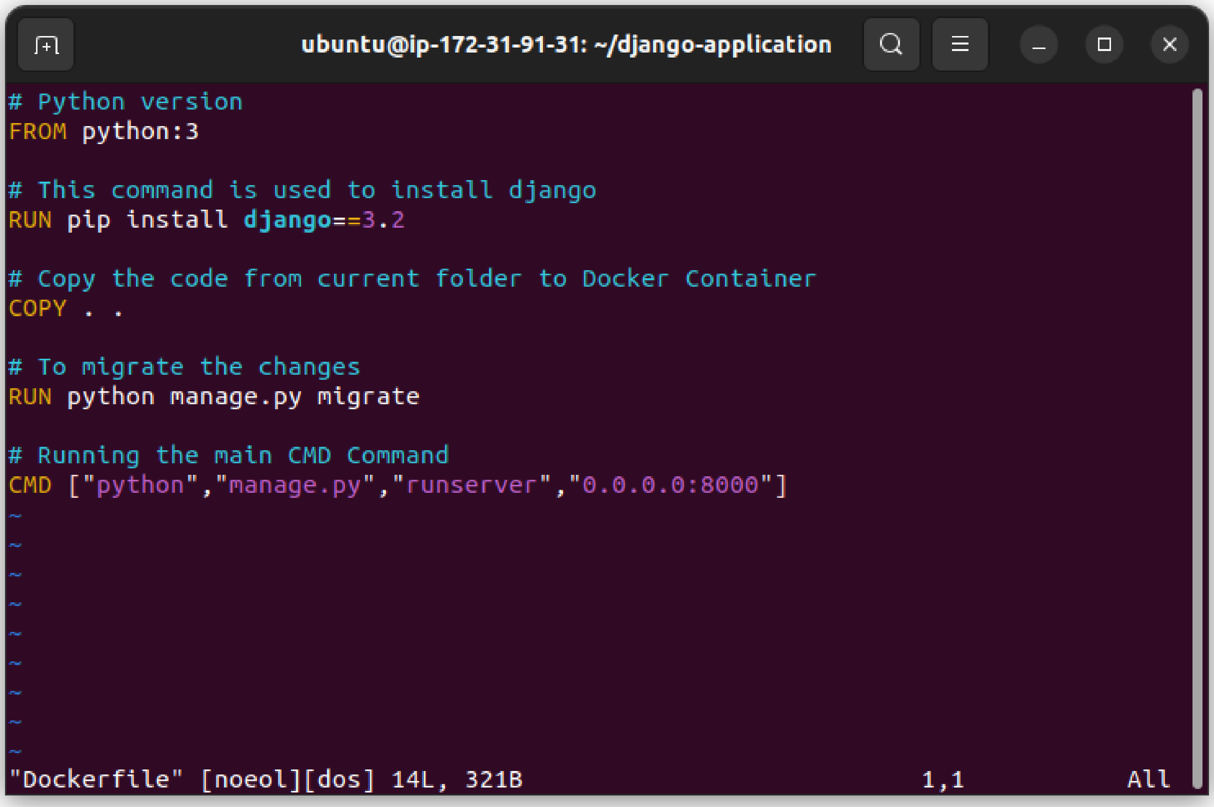
```
ssh -i "projects.pem" ubuntu@ec2-54-196-43-216.compute-1.amazonaws.com
```

Note: In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

3. Go to the download folder, where the .pem file is placed and open the terminal in the same location, and paste the SSH.

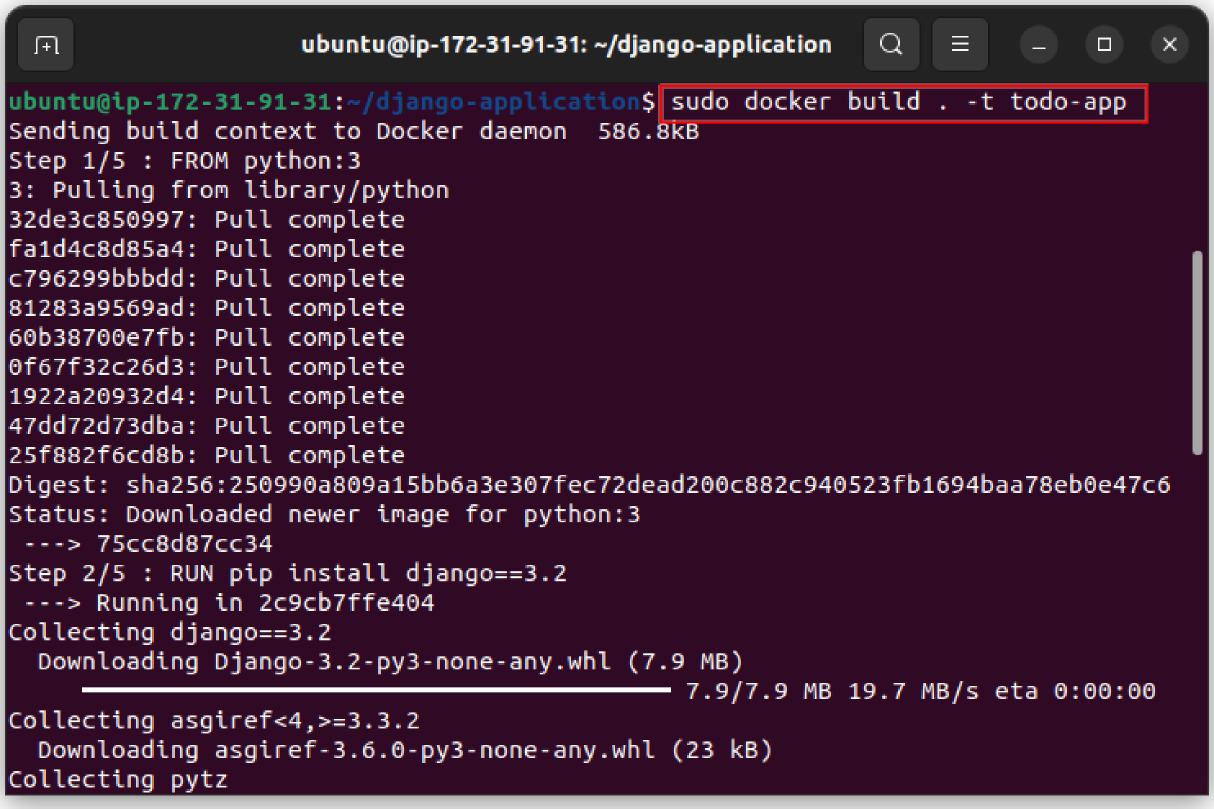
4. No we will Install Docker. By running this Command:
“Sudo apt-install docker.io”

5. Write Docker file for running you application, and Push your Docker file in Project Repo on Github.

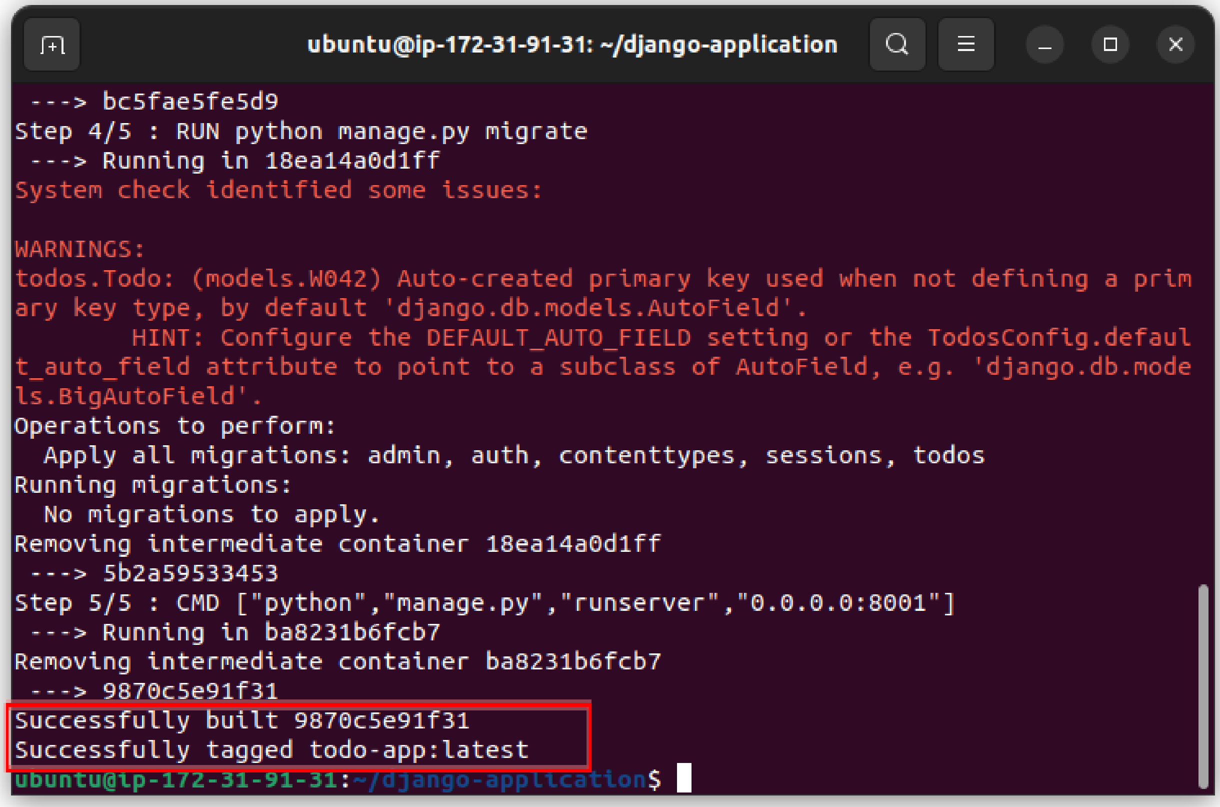

ubuntu@ip-172-31-91-31: ~/django-application

```
# Python version
FROM python:3

# This command is used to install django
RUN pip install django==3.2

# Copy the code from current folder to Docker Container
COPY . .

# To migrate the changes
RUN python manage.py migrate

# Running the main CMD Command
CMD ["python","manage.py","runserver","0.0.0.0:8000"]
~
~
~
~
~
~
~
~
~
"Dockerfile" [noeol][dos] 14L, 321B                          1,1           All
```

6. Build image of Docker file by running this Command:
It will give some warning but it will docker image of your application.
“sudo docker build . -t todo-app”



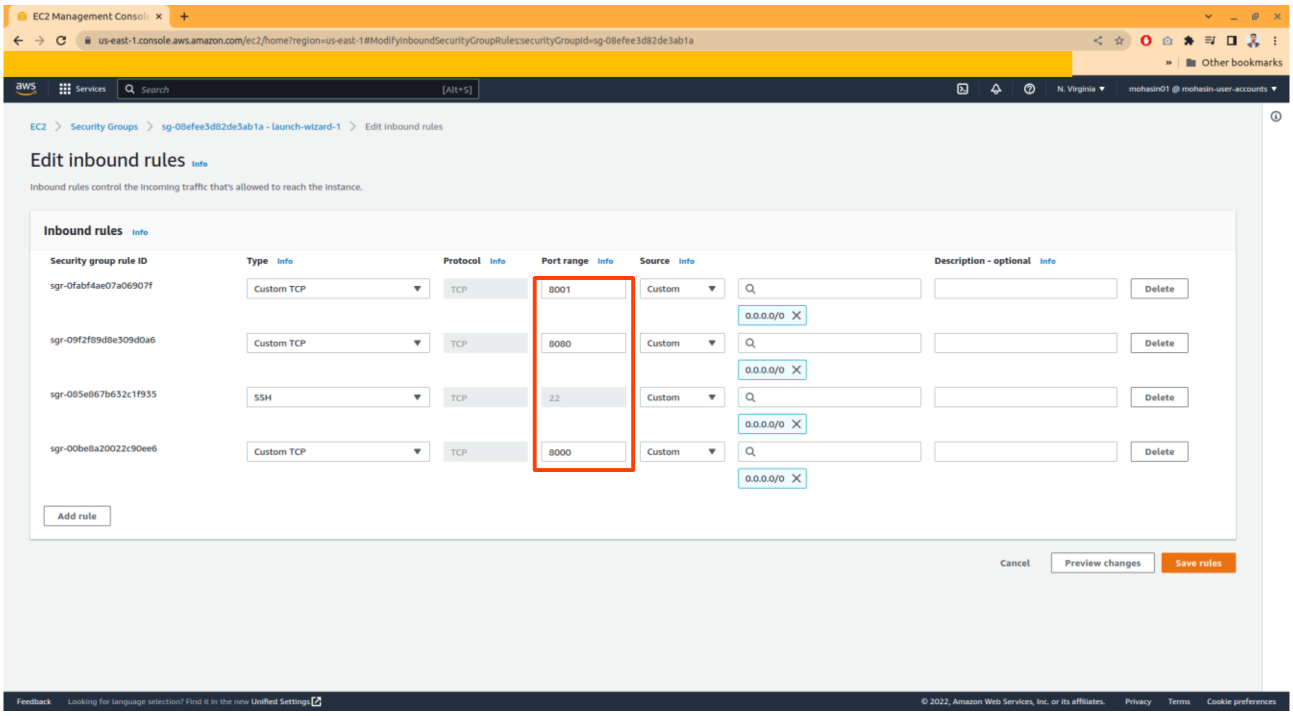
```
ubuntu@ip-172-31-91-31:~/django-application$ sudo docker build . -t todo-app
Sending build context to Docker daemon  586.8kB
Step 1/5 : FROM python:3
3: Pulling from library/python
32de3c850997: Pull complete
fa1d4c8d85a4: Pull complete
c796299bbbdd: Pull complete
81283a9569ad: Pull complete
60b38700e7fb: Pull complete
0f67f32c26d3: Pull complete
1922a20932d4: Pull complete
47dd72d73dba: Pull complete
25f882f6cd8b: Pull complete
Digest: sha256:250990a809a15bb6a3e307fec72dead200c882c940523fb1694baa78eb0e47c6
Status: Downloaded newer image for python:3
 ---> 75cc8d87cc34
Step 2/5 : RUN pip install django==3.2
 ---> Running in 2c9cb7ffe404
Collecting django==3.2
  Downloading Django-3.2-py3-none-any.whl (7.9 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.9/7.9 MB 19.7 MB/s eta 0:00:00
Collecting asgiref<4,>=3.3.2
  Downloading asgiref-3.6.0-py3-none-any.whl (23 kB)
Collecting pytz
```



```
 ---> bc5fae5fe5d9
Step 4/5 : RUN python manage.py migrate
 ---> Running in 18ea14a0d1ff
System check identified some issues:

WARNINGS:
todos.Todo: (models.W042) Auto-created primary key used when not defining a prim
ary key type, by default 'django.db.models.AutoField'.
        HINT: Configure the DEFAULT_AUTO_FIELD setting or the TodosConfig.defaul
t_auto_field attribute to point to a subclass of AutoField, e.g. 'django.db.mode
ls.BigAutoField'.
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, todos
Running migrations:
  No migrations to apply.
Removing intermediate container 18ea14a0d1ff
 ---> 5b2a59533453
Step 5/5 : CMD ["python","manage.py","runserver","0.0.0.0:8001"]
 ---> Running in ba8231b6fcb7
Removing intermediate container ba8231b6fcb7
 ---> 9870c5e91f31
Successfully built 9870c5e91f31
Successfully tagged todo-app:latest
ubuntu@ip-172-31-91-31:~/django-application$ 
```

7. Now we will install Jenkins on the machine, by following this link https://www.jenkins.io/doc/book/installing/linux/

8. Now, we will allow ports 8080 and 8001 or 8000 (on which you want to run your application) for this machine from a security group. We can find the security group in the EC2 description. Now, here we need to allow "Inbound Rule" as below:


EC2 Management Console
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#ModifyInboundSecurityGroupRules:securityGroupId=sg-08efee3d82de3ab1a
Other bookmarks
Services
Search
[Alt+S]
N. Virginia
mohasin01 @ mohasin-user-accounts
EC2 > Security Groups > sg-08efee3d82de3ab1a - launch-wizard-1 > Edit inbound rules

Edit inbound rules

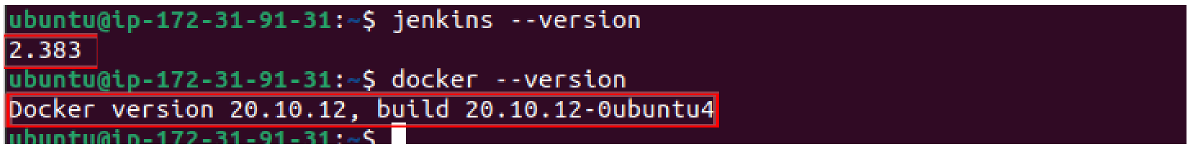
Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules

Security group rule ID	Type	Protocol	Port range	Source		Description - optional	
sgr-0fabf4ae07a06907f	Custom TCP	TCP	8001	Custom	0.0.0.0/0		Delete
sgr-09f2f89d8e309d0a6	Custom TCP	TCP	8080	Custom	0.0.0.0/0		Delete
sgr-085e867b632c1f935	SSH	TCP	22	Custom	0.0.0.0/0		Delete
sgr-00be8a20022c90ee6	Custom TCP	TCP	8000	Custom	0.0.0.0/0		Delete

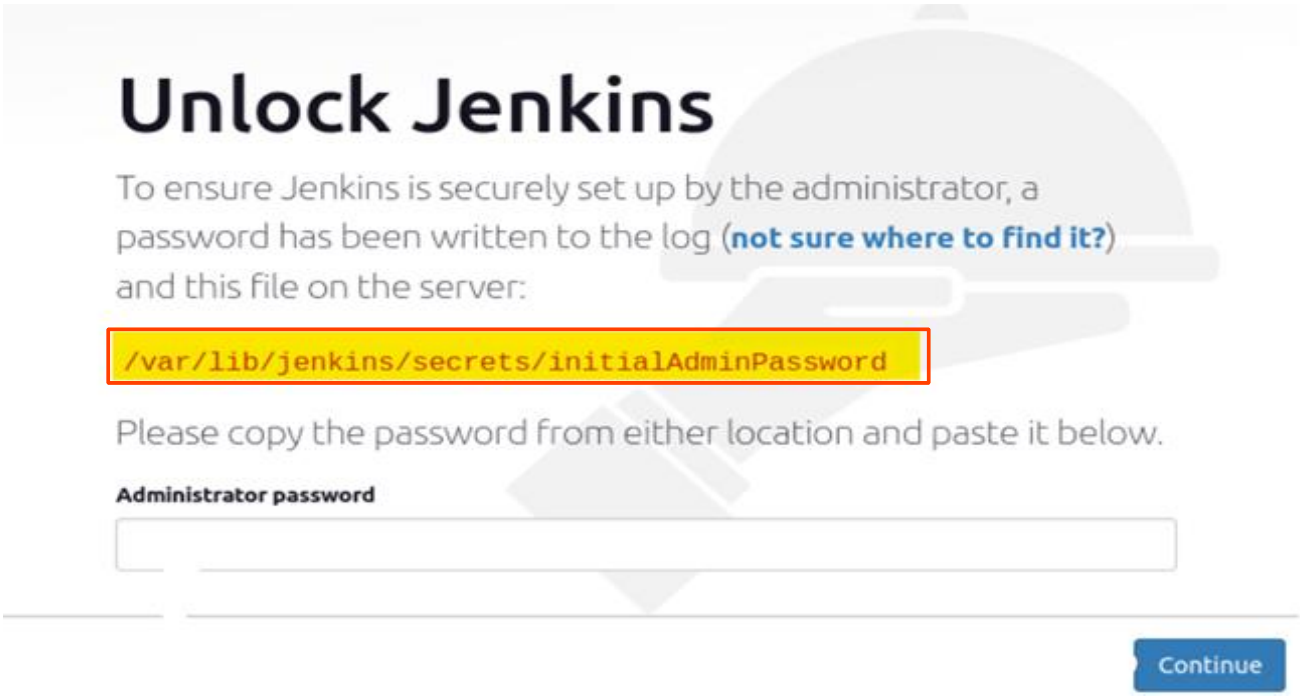
Add rule
Cancel
Preview changes
Save rules
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

9. Now check if it got installed by running "**jenkins --version**" and "**docker –version**"



```
ubuntu@ip-172-31-91-31:~$ jenkins --version
2.383
ubuntu@ip-172-31-91-31:~$ docker --version
Docker version 20.10.12, build 20.10.12-0ubuntu4
ubuntu@ip-172-31-91-31:~$
```

10. Now, Copy the Public Ip of the machine and paste it to the browser to access the Jenkins portal with the port no **8080**. (As your Jenkins will Run on Port 8080).
"Public Ip of EC2 :8080"



Unlock Jenkins

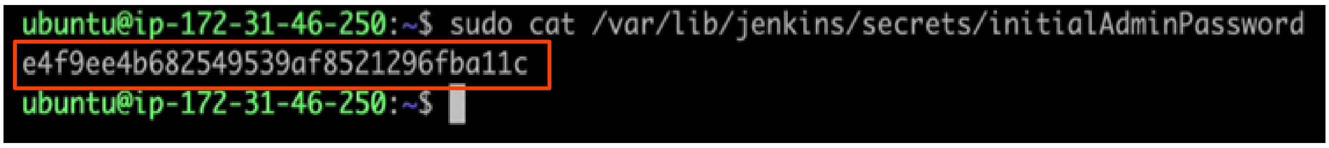
To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

/var/lib/jenkins/secrets/initialAdminPassword

Please copy the password from either location and paste it below.

Administrator password

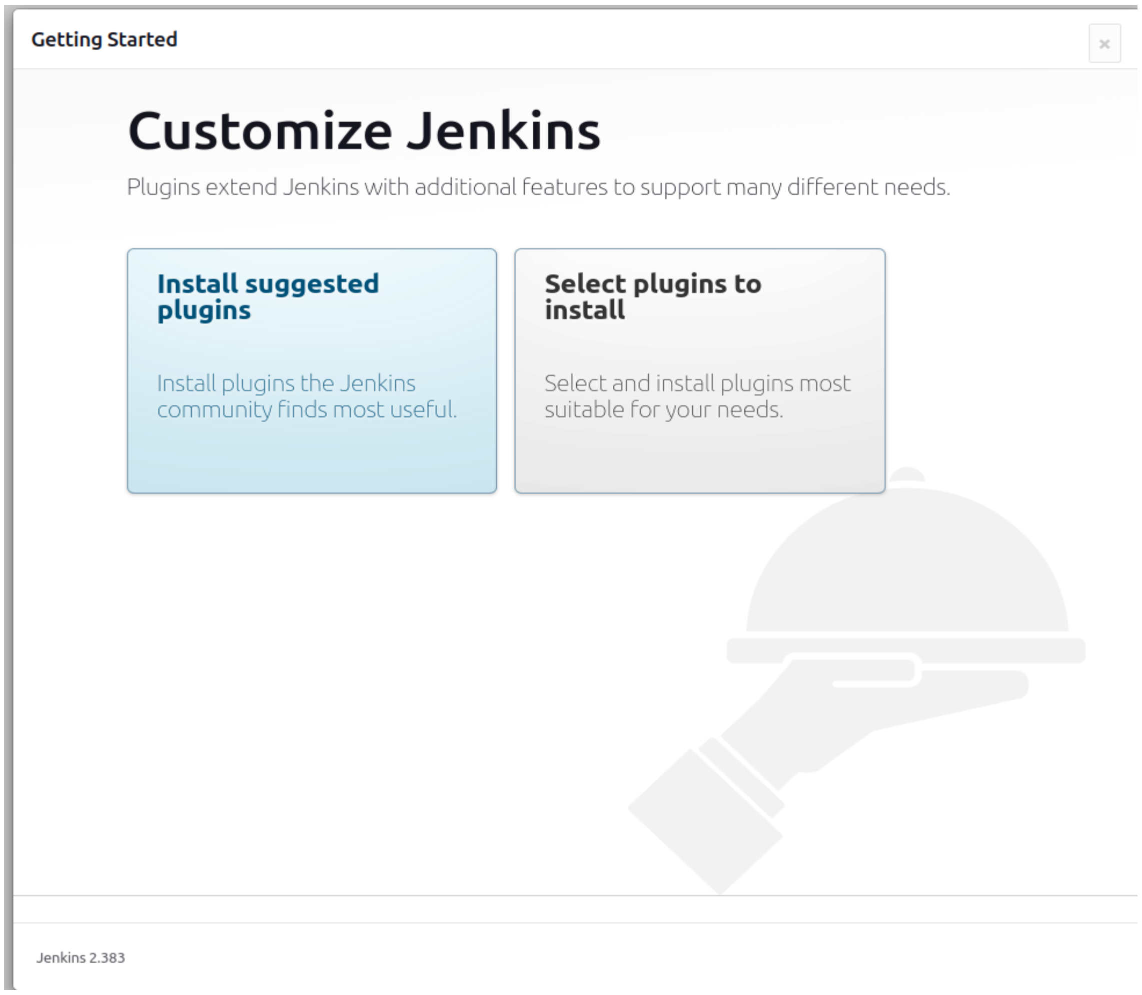
11. We need an Administrator Password to unlock this. For that, go to the provided highlighted path in the upper screenshot.
"cat /var/lib/Jenkins/secrets/initialAdminPassword"



```
ubuntu@ip-172-31-46-250:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
e4f9ee4b682549539af8521296fba11c
ubuntu@ip-172-31-46-250:~$
```

Paste this password in the "**Administrator Password**" Column and Continue.

12. Now Click on, "**Install Suggested Plugins**"


Getting Started

Customize Jenkins

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

Install suggested plugins

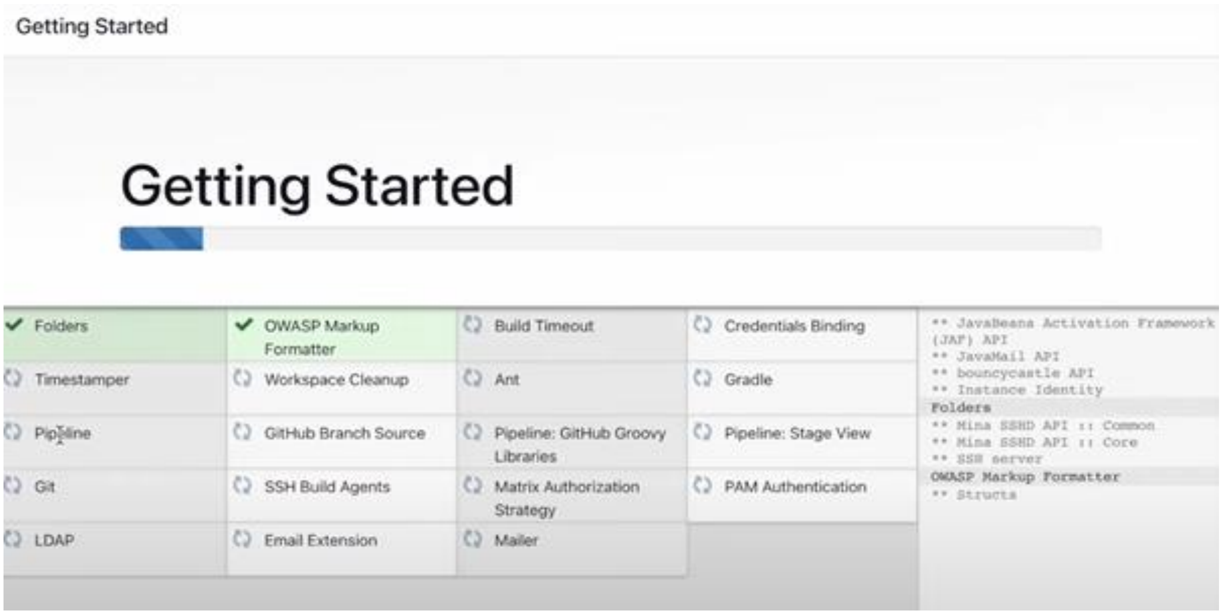
Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

Jenkins 2.383

13. This will now install all the basic plugins that you will need mostly.

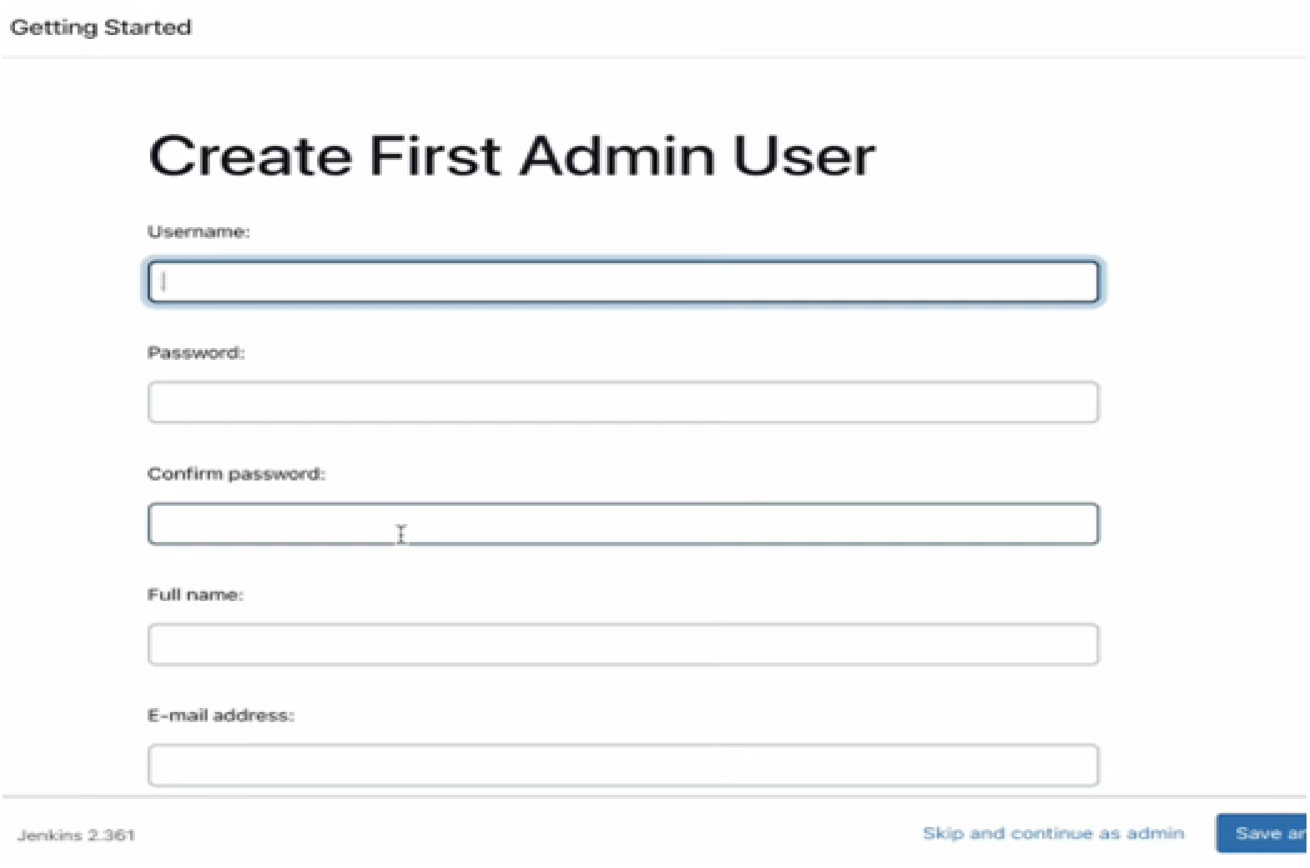

Getting Started

Getting Started

✓ Folders	✓ OWASP Markup Formatter	Build Timeout	Credentials Binding
Timestamper	Workspace Cleanup	Ant	Gradle
Pipeline	GitHub Branch Source	Pipeline: GitHub Groovy Libraries	Pipeline: Stage View
Git	SSH Build Agents	Matrix Authorization Strategy	PAM Authentication
LDAP	Email Extension	Mailer	

```
** JavaBeans Activation Framework
(JAF) API
** JavaMail API
** bouncycastle API
** Instance Identity
Folders
** Mina SSHD API :: Common
** Mina SSHD API :: Core
** SSH server
OWASP Markup Formatter
** Structs
```

14. Now, Jenkins will ask us to create the **First Admin User**.


Getting Started

Create First Admin User

Username:

Password:

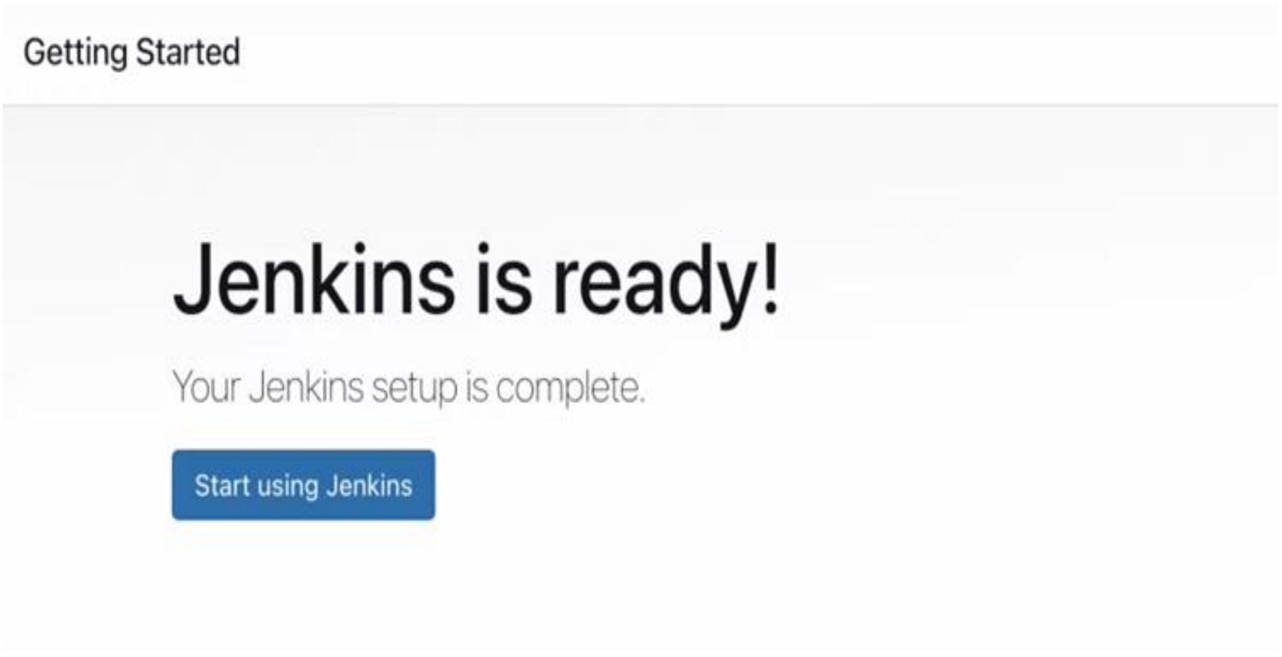
Confirm password:

Full name:

E-mail address:

Jenkins 2.361

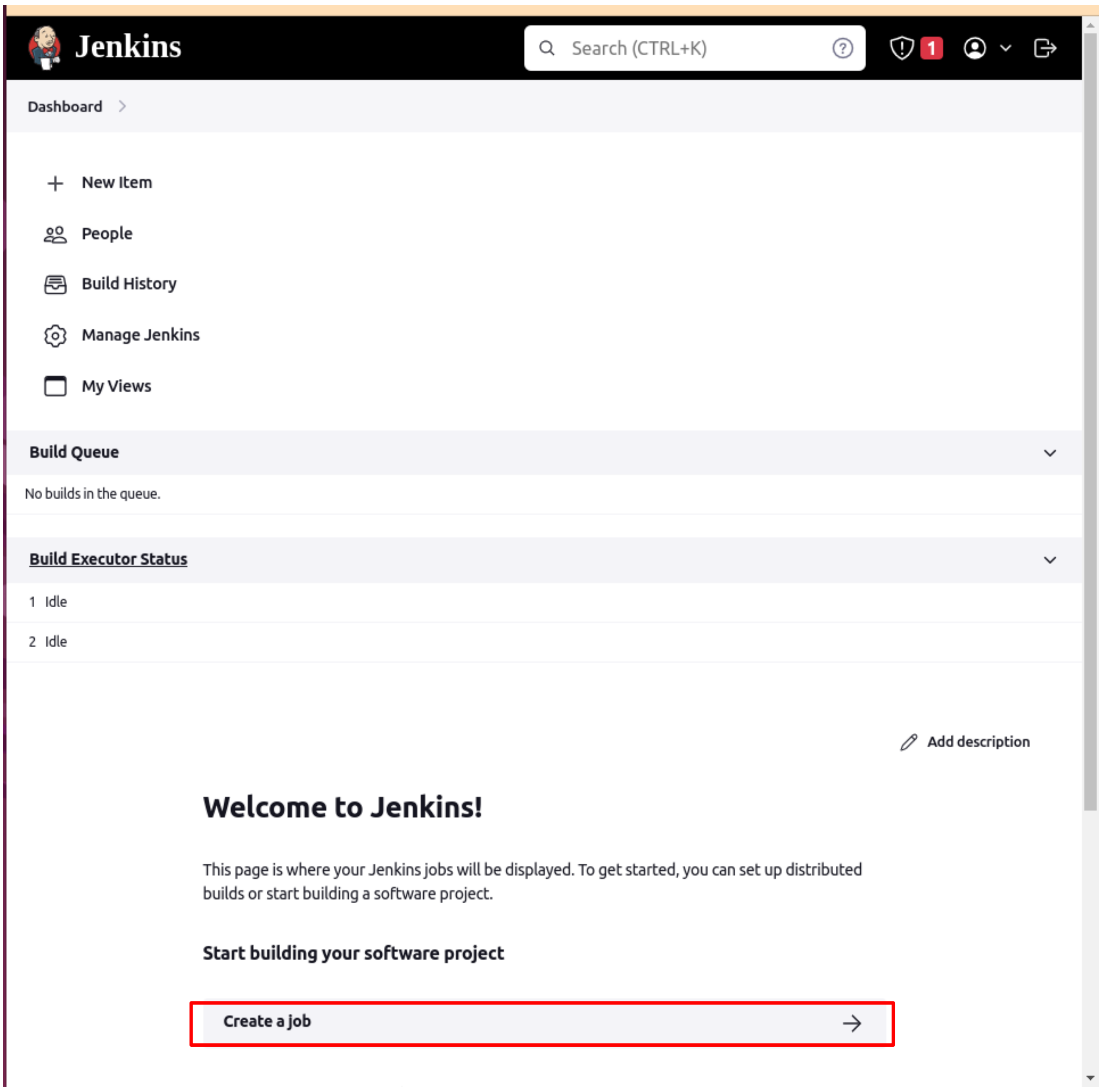
15. Add the fields according to your username and Email.


Getting Started

Jenkins is ready!

Your Jenkins setup is complete.

16. The Jenkins homepage will look like this,
First of all we **create a job**.



Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

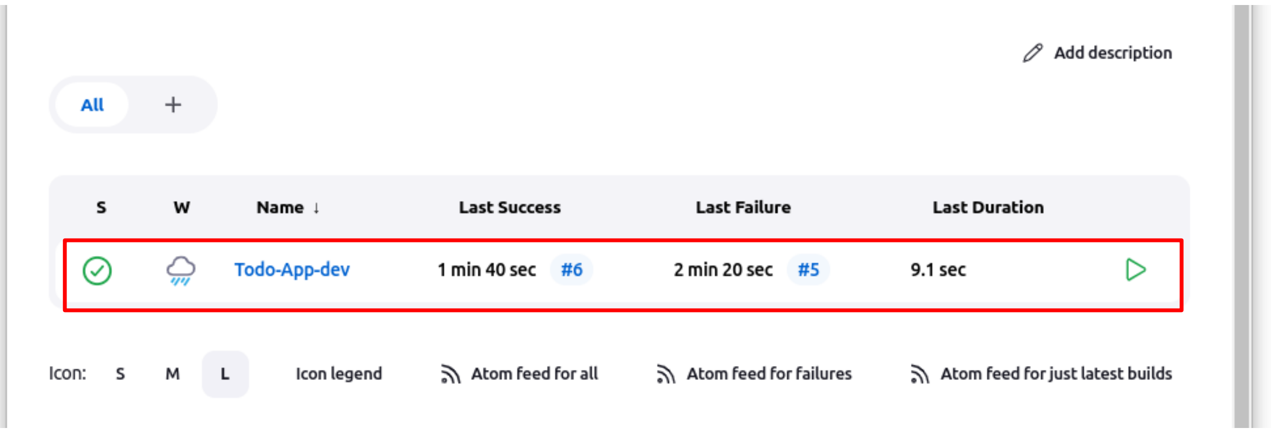
Add description

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

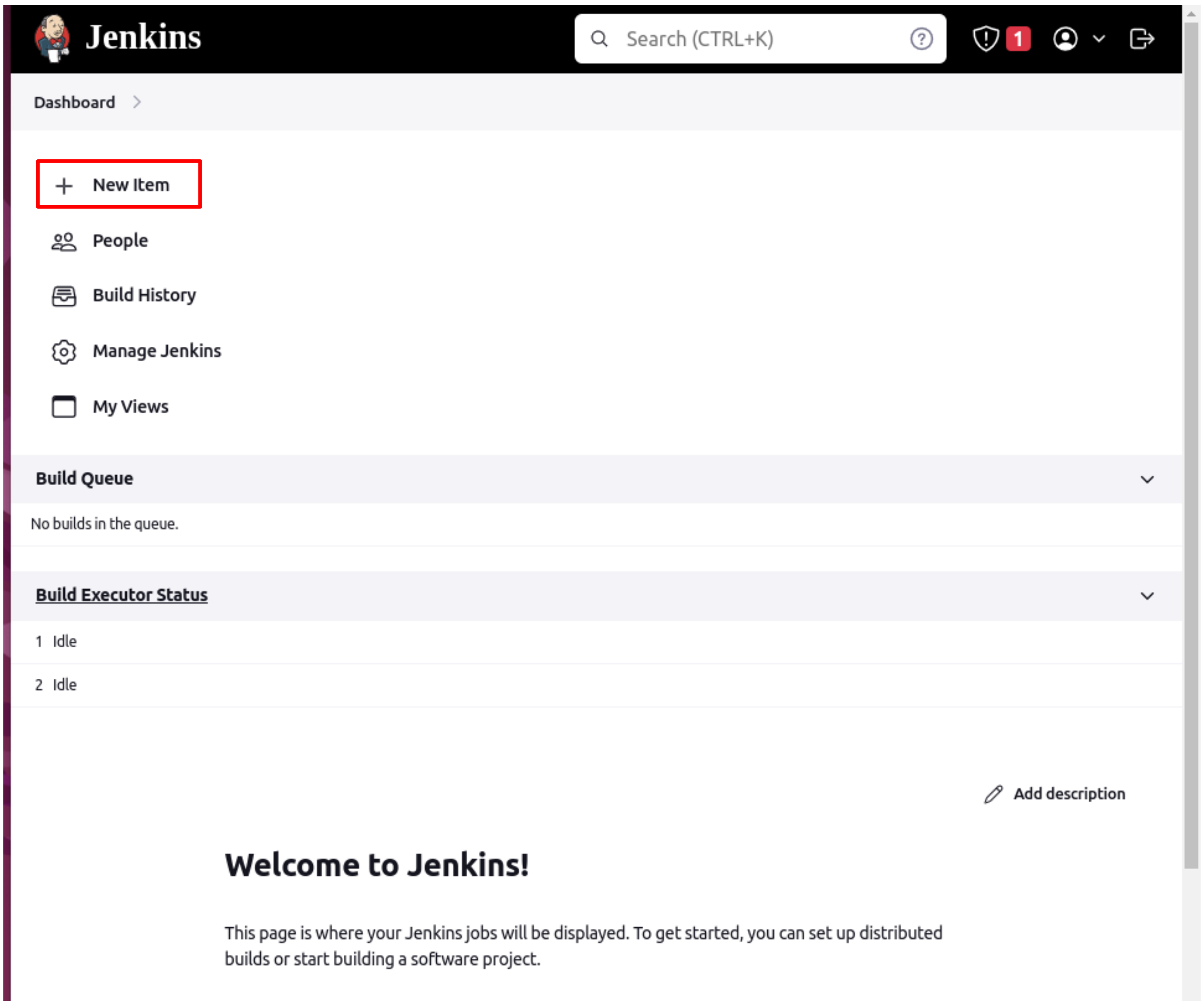
Create a job



S	W	Name ↓	Last Success	Last Failure	Last Duration
✓		Todo-App-dev	1 min 40 sec #6	2 min 20 sec #5	9.1 sec

17. Now, we will create a **CI/CD pipeline**, which will **fetch the code from GitHub**.

18. From Jenkins Dashboard, Click on “**New Item**”.



No builds in the queue.

1 Idle

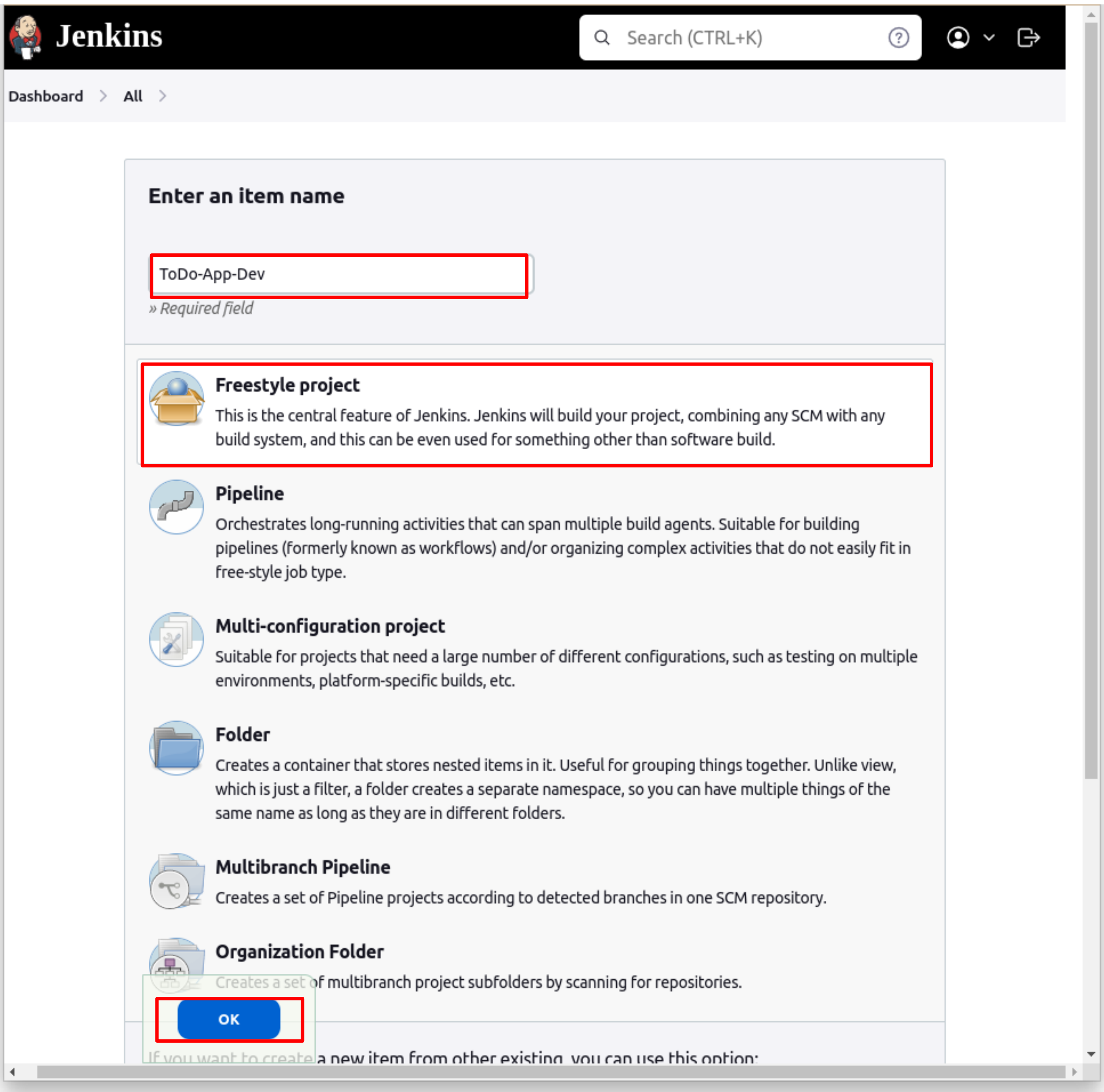
2 Idle

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

19. Now, Add name as

- Name: **ToDo-App-Dev**
- Project: **Freestyle project**
- Click “Ok”.



Enter an item name

ToDo-App-Dev

» 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.

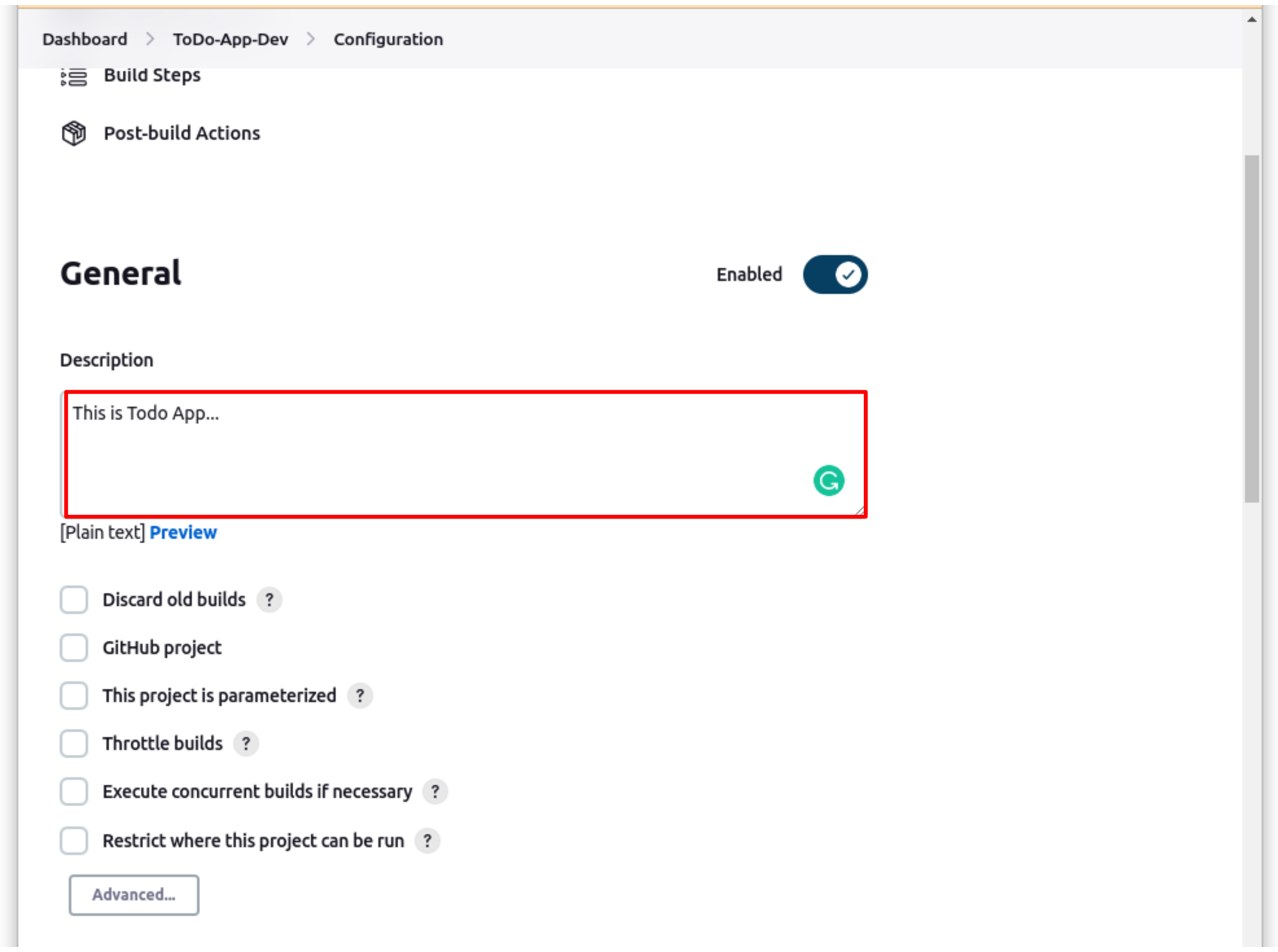
Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

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:

20. Here, we need to fill up the description.



Dashboard > ToDo-App-Dev > Configuration

Build Steps

Post-build Actions

General

Enabled

Description

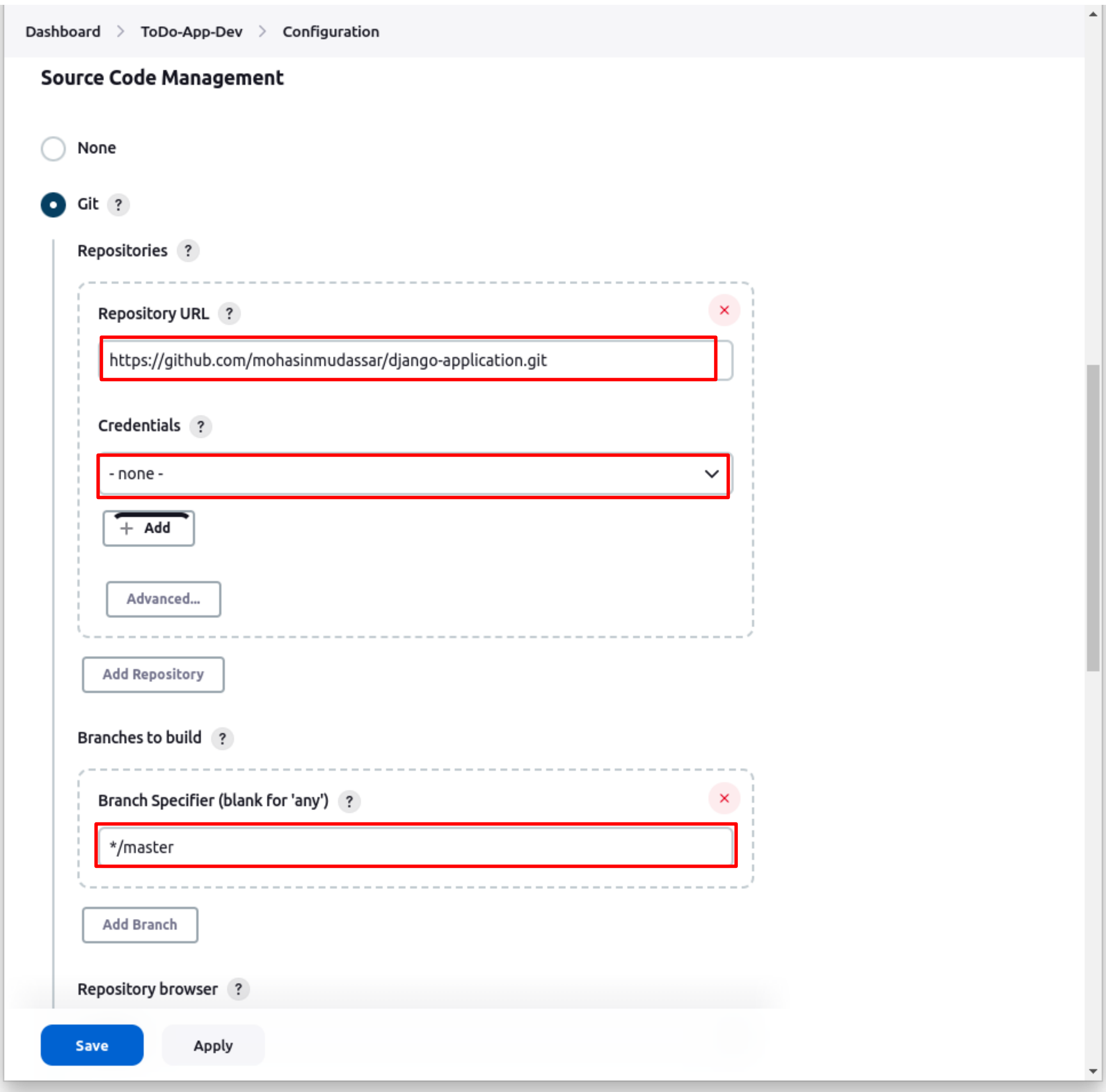
This is Todo App...

[Plain text] Preview

- [ ] Discard old builds
- [ ] GitHub project
- [ ] This project is parameterized
- [ ] Throttle builds
- [ ] Execute concurrent builds if necessary
- [ ] Restrict where this project can be run

Advanced...

21. In Source Code Management, select Git and **Add Repository URL and Credentials.** (If there is not any added credential, we need to add). I have addes them from configure so it will fetch it from there. You can also add the branch


Dashboard > ToDo-App-Dev > Configuration

Source Code Management

None

Git

Repositories

Repository URL: https://github.com/mohasinmudassar/django-application.git

Credentials: - none -

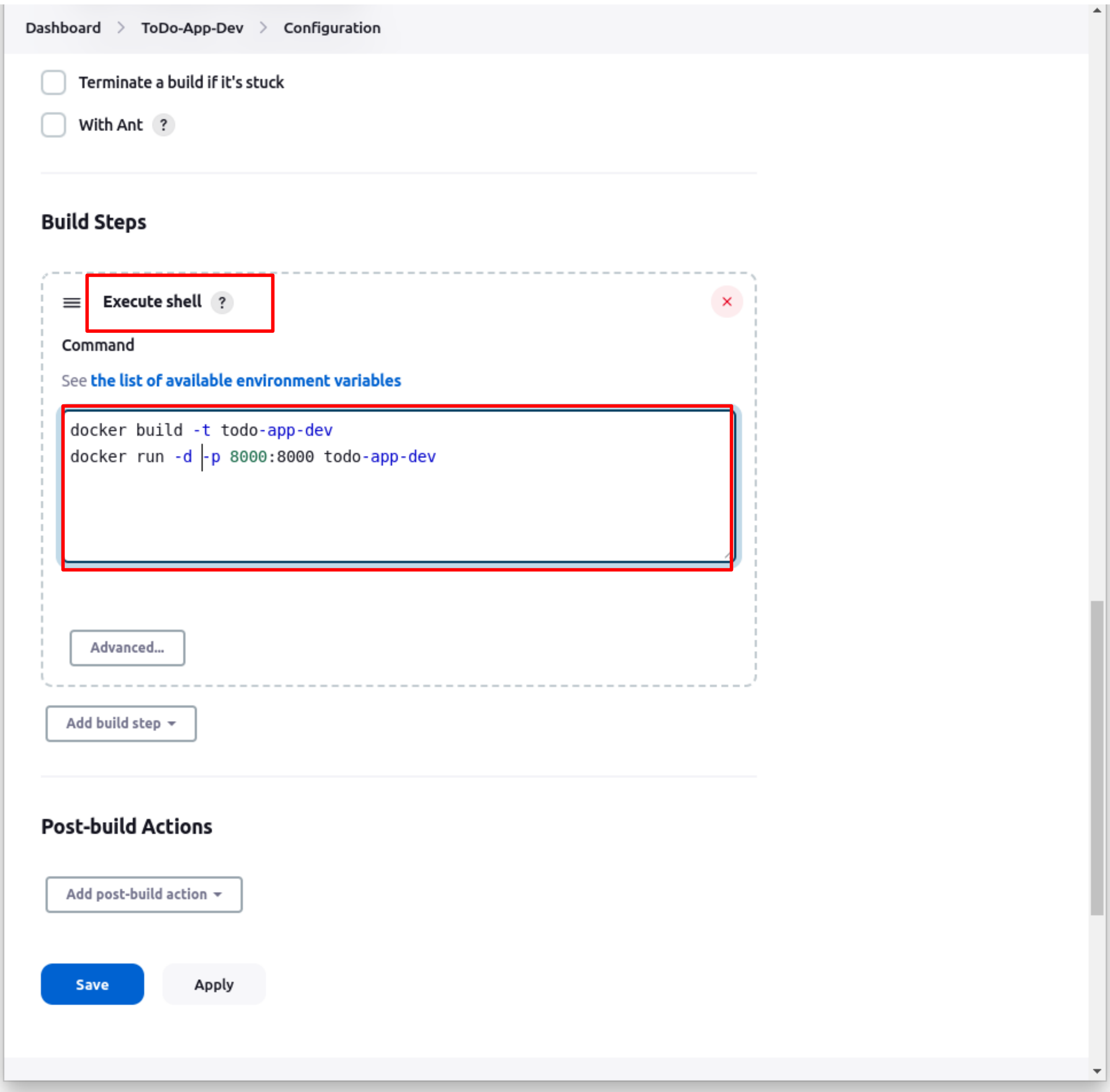
Branches to build

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

Repository browser

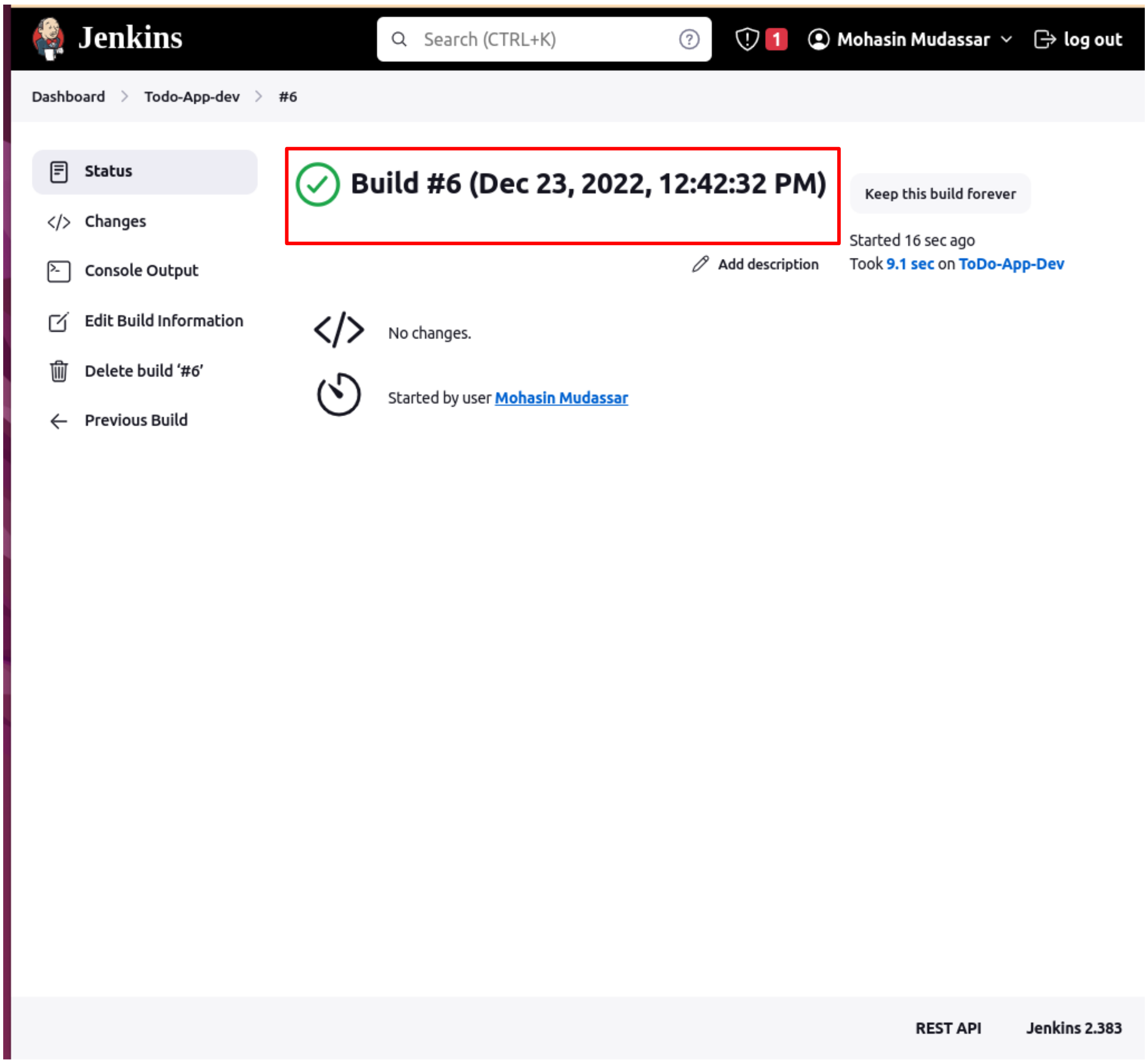
22. Build Step, select Execute Shell and write following command to build docker image and from Docker image we will create a container.

“docker build -t todo-app-dev ”
“docker run -d -p 8000:8000 todo-app-dev “


Dashboard > ToDo-App-Dev > Configuration
Terminate a build if it's stuck
With Ant
Build Steps
Execute shell
Command
See the list of available environment variables
Advanced...
Add build step
Post-build Actions
Add post-build action
Save
Apply

```
docker build -t todo-app-dev
docker run -d -p 8000:8000 todo-app-dev
```

23. Now, Click on build Now. And the build will be started, in the build history.



Jenkins

Search (CTRL+K)

Mohasin Mudassar

log out

Dashboard > Todo-App-dev > #6

Status

Changes

Console Output

Edit Build Information

Delete build '#6'

Previous Build

Build #6 (Dec 23, 2022, 12:42:32 PM)

Keep this build forever

Started 16 sec ago

Took 9.1 sec on ToDo-App-Dev

Add description

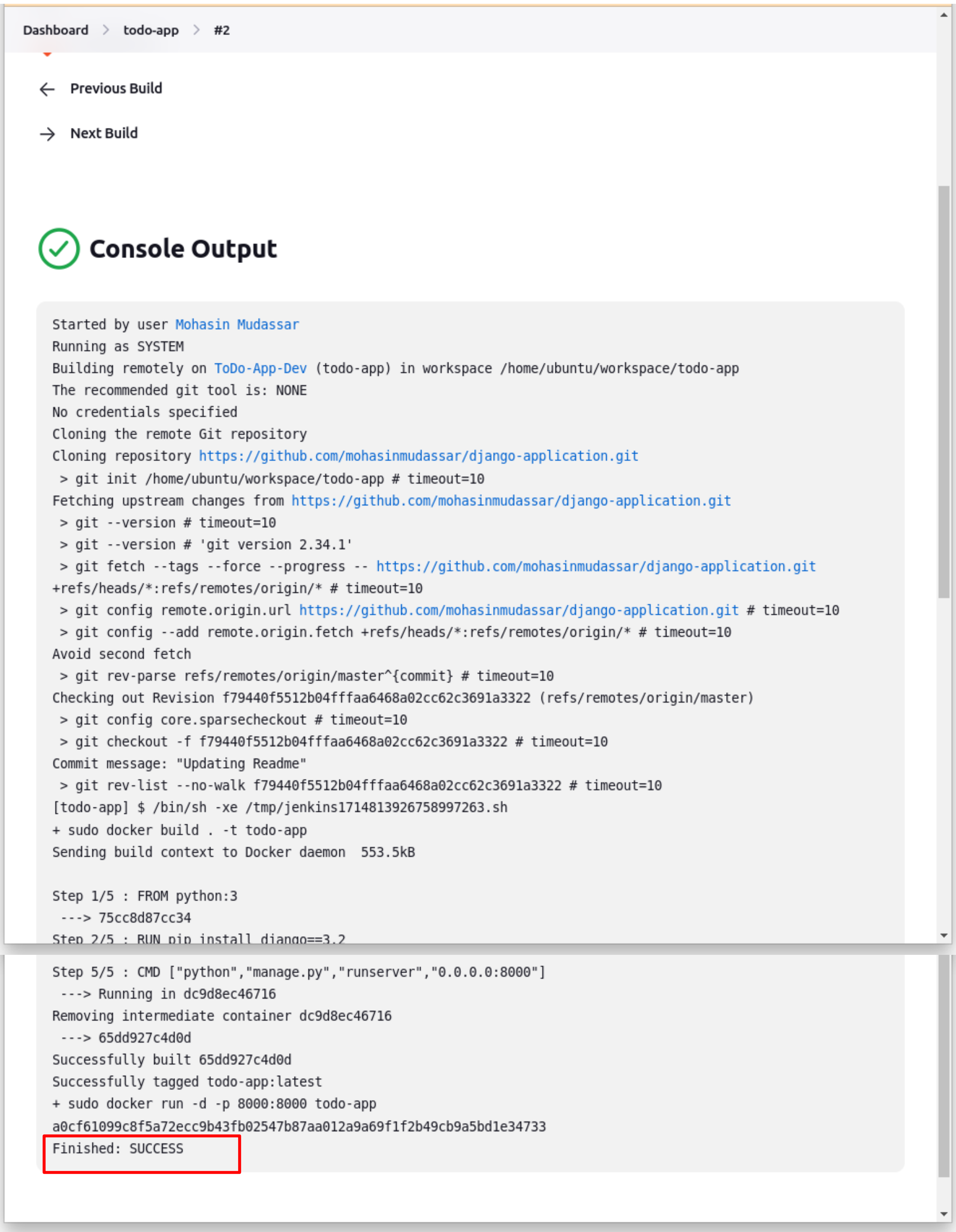
No changes.

Started by user Mohasin Mudassar

REST API

Jenkins 2.383

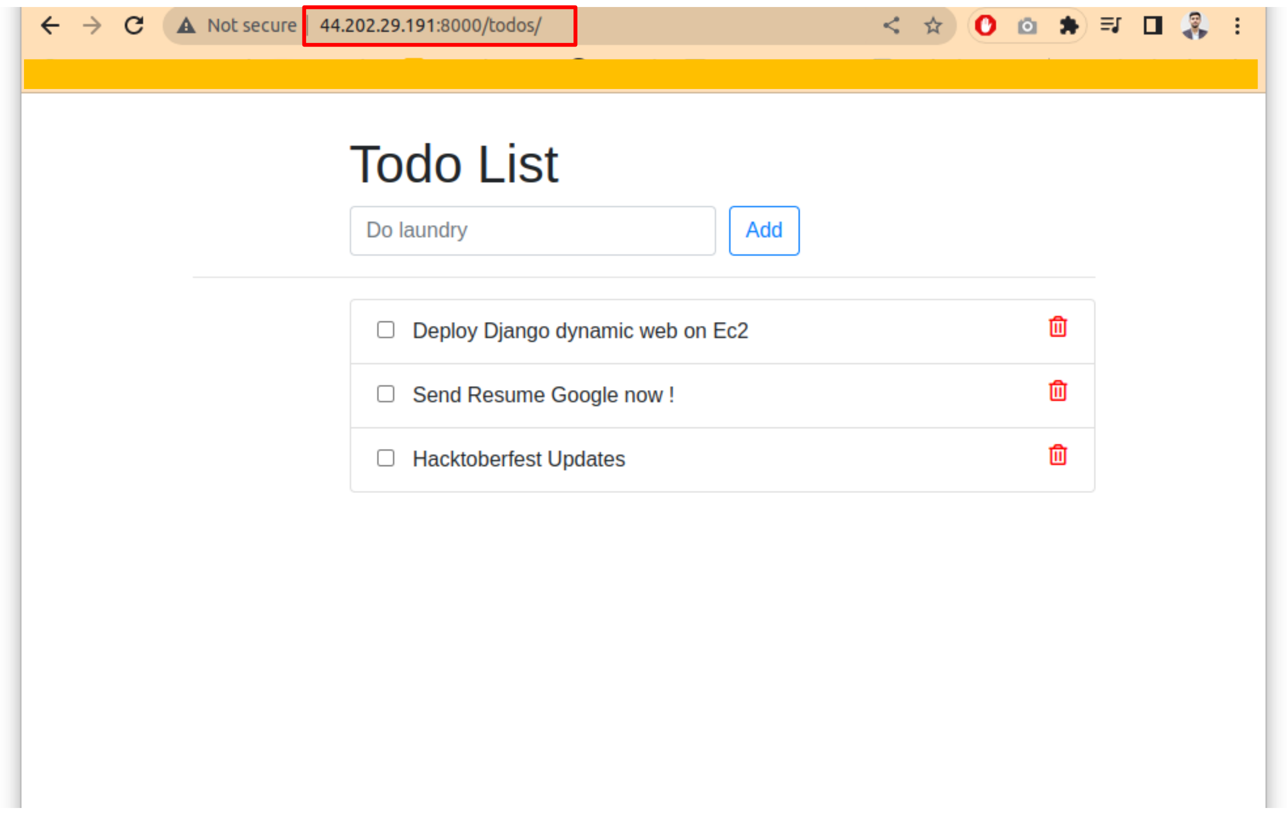
24. We can check Output Console for any error, **SUCCESS**.



```
Started by user Mohasin Mudassar
Running as SYSTEM
Building remotely on ToDo-App-Dev (todo-app) in workspace /home/ubuntu/workspace/todo-app
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/mohasinmudassar/django-application.git
 > git init /home/ubuntu/workspace/todo-app # timeout=10
Fetching upstream changes from https://github.com/mohasinmudassar/django-application.git
 > git --version # timeout=10
 > git --version # 'git version 2.34.1'
 > git fetch --tags --force --progress -- https://github.com/mohasinmudassar/django-application.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
 > git config remote.origin.url https://github.com/mohasinmudassar/django-application.git # timeout=10
 > git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision f79440f5512b04fffaa6468a02cc62c3691a3322 (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f f79440f5512b04fffaa6468a02cc62c3691a3322 # timeout=10
Commit message: "Updating Readme"
 > git rev-list --no-walk f79440f5512b04fffaa6468a02cc62c3691a3322 # timeout=10
[todo-app] $ /bin/sh -xe /tmp/jenkins1714813926758997263.sh
+ sudo docker build . -t todo-app
Sending build context to Docker daemon  553.5kB

Step 1/5 : FROM python:3
 ---> 75cc8d87cc34
Step 2/5 : RUN pip install django==3.2
```

```
Step 5/5 : CMD ["python","manage.py","runserver","0.0.0.0:8000"]
 ---> Running in dc9d8ec46716
Removing intermediate container dc9d8ec46716
 ---> 65dd927c4d0d
Successfully built 65dd927c4d0d
Successfully tagged todo-app:latest
+ sudo docker run -d -p 8000:8000 todo-app
a0cf61099c8f5a72ecc9b43fb02547b87aa012a9a69f1f2b49cb9a5bd1e34733
Finished: SUCCESS
```

25. After getting success, In the browser, search for “public_ip_of_ec2:8000”



Deploy Django dynamic web on Ec2

Send Resume Google now !

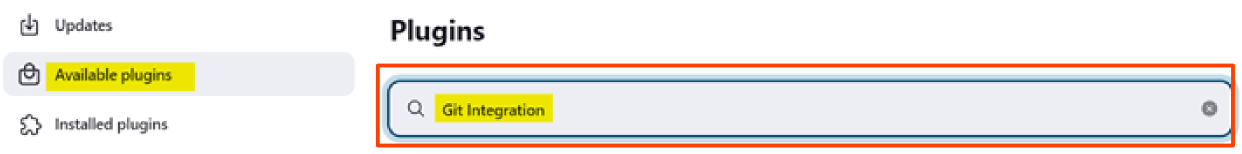
Hacktoberfest Updates

Now, our goal is,
·Whenever the developer commits their code in GitHub, after every commit, it should reflect in the live web app.
·For that, we will use “**GitScm polling**”.
·Every time, a developer made a commit, a trigger will run automatically, which will rebuild the image and run a container on your behalf as a part of automation that will run the pipeline automatically.

26. Now, configure the project again, and add

- Build Trigger: **GitHub hook trigger for GitScm polling**.
- Description: **GitHub webhook integration**.

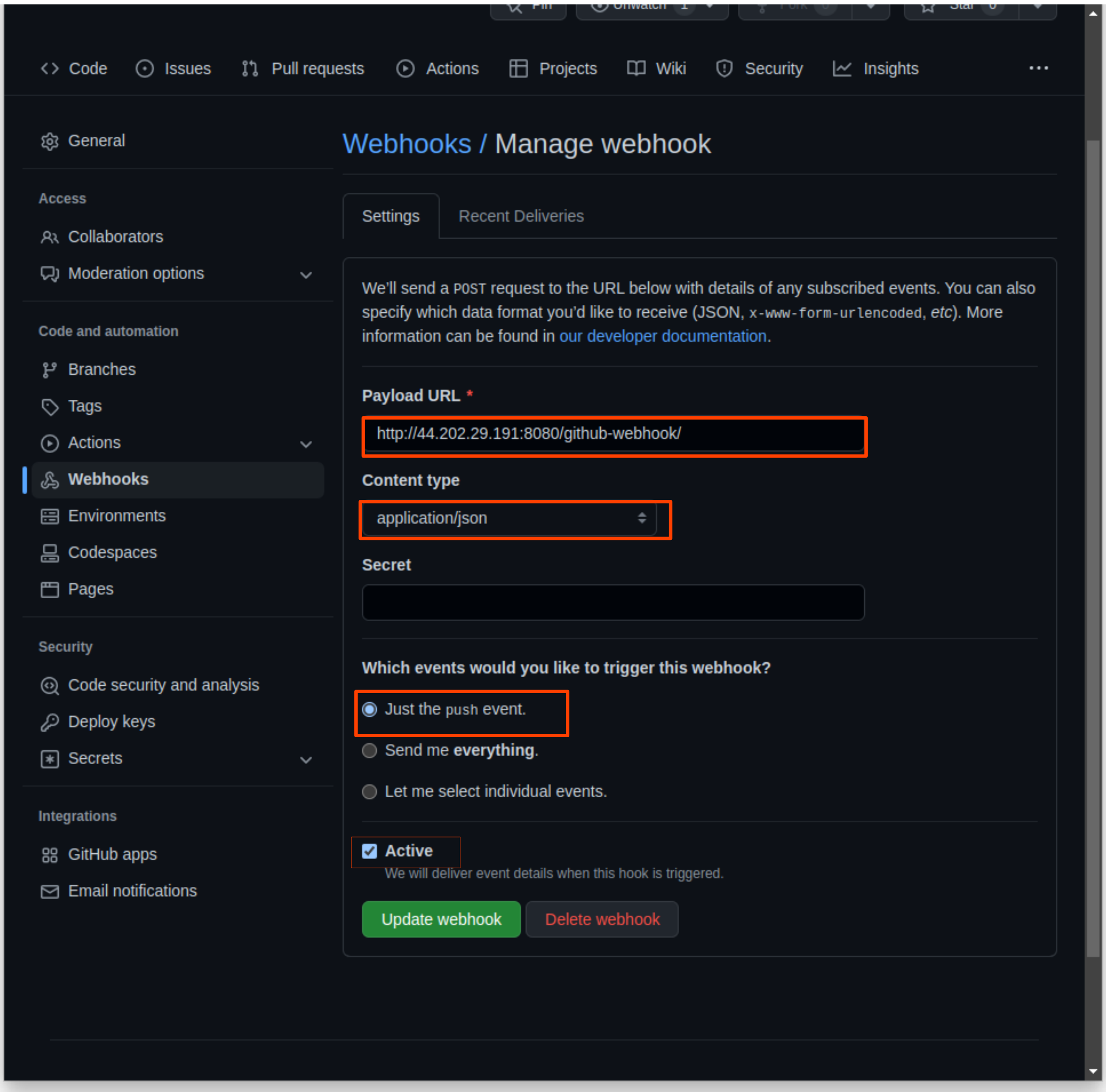
27. We need to install the “Git Integration” plugin from Manage Jenkins, by following the path,
(Manage Jenkins > Manage Plugins > Git Integration).



28. Now, we need to go to GitHub and create a Webhook.
GitHub > Your Project Repository> Settings > Webhooks

29. Add the following details,

- Payload URL: **http://<public_ip_of_ec2>:8080/github-webhook/**
- Content Type: **application/json**
- Which event would you like to trigger this webhook?
- Just the **push event.**
- Active: True
- Click on "**Add Webhook**".



Webhooks / Manage webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, *etc*). More information can be found in our developer documentation.

Payload URL *: http://44.202.29.191:8080/github-webhook/

Content type: application/json

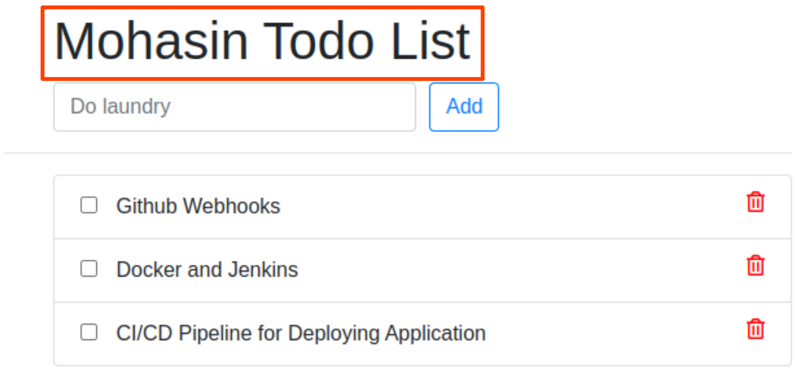
Secret:

Which events would you like to trigger this webhook?

- [✓ Just the push event.]
- Send me **everything**.
- Let me select individual events.

[✓ Active]
We will deliver event details when this hook is triggered.

30. Do some changes in the code and push it to GitHub, this will automatically run a pipeline, and the new code will be Live.
After Webhook Deploying.



Mohasin Todo List

Do laundry

Add

- Github Webhooks
- Docker and Jenkins
- CI/CD Pipeline for Deploying Application

Resources:
Project Github Link:
https://github.com/mohasinmudassar/django-application.git