**Project 1 — DevOps (Implementation)**

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

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

Follow the steps:

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

· Name: jenkins-server

· AMI: ubuntu.

· Instance type: t2.micro (free tier).

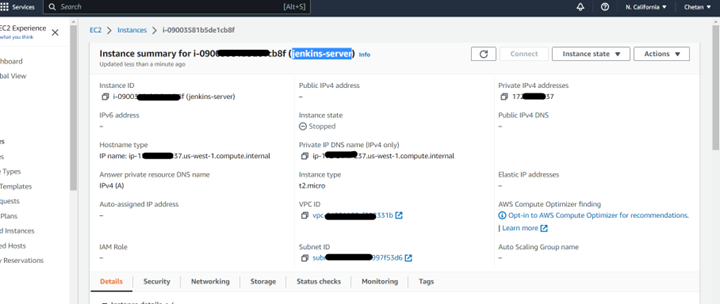
· Key pair login: Create > docker.pem.

· Allow HTTP.

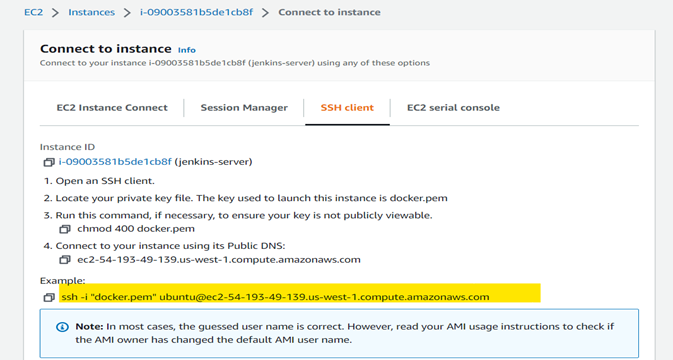
· Allow HTTPS.

(Download the .pem file.)

Click on Launch Instance.



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



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. In the machine, run the command

“ssh-keygen”

This will generate public and private keys in the machine.

Id\_rsa — Private Key.

Id\_rsa.pub — Public Key.



4. In the machine, run the command “ssh-keygen” This will generate public and private keys in the machine. Id\_rsa — Private Key. Id\_rsa.pub — Public Key.

ChatGPT

The ssh-keygen command is used to generate a pair of public and private keys for secure communication over SSH (Secure Shell). When you run the command:

bash

ssh-keygen

It will prompt you for some information and then generate the keys in the default location, typically in the ~/.ssh/ directory. The keys generated will be:

1. **Private Key**: id\_rsa - This key should be kept secret and never shared. It is used to decrypt messages that were encrypted with the corresponding public key.
2. **Public Key**: id\_rsa.pub - This key can be shared with others. It is used by other parties to encrypt messages that can only be decrypted by the private key.

After running the ssh-keygen command, you may need to provide a passphrase for additional security. This passphrase adds an extra layer of protection to your private key.

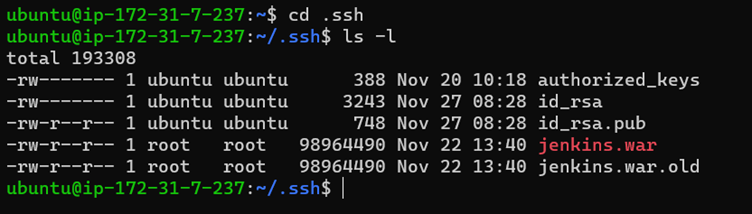
Here's a brief overview of what each key is used for:

* **Private Key (id\_rsa):** This key should be kept secure on your local machine. It's used to decrypt messages that were encrypted with the corresponding public key.
* **Public Key (id\_rsa.pub):** This key can be shared with others, for example, with a server you want to connect to securely. The server will use this public key to encrypt messages that can only be decrypted by your private key.

Remember to handle your private key with care and never share it with anyone you don't trust. The public key, on the other hand, can be freely distributed to allow others to establish secure communication with you.

Top of Form

Bottom of Form



5. Now we will install Jenkins on the machine, by following this link

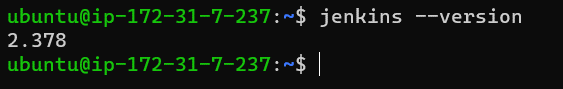
<https://www.jenkins.io/doc/book/installing/linux/>

This will automatically install java with Jenkins.

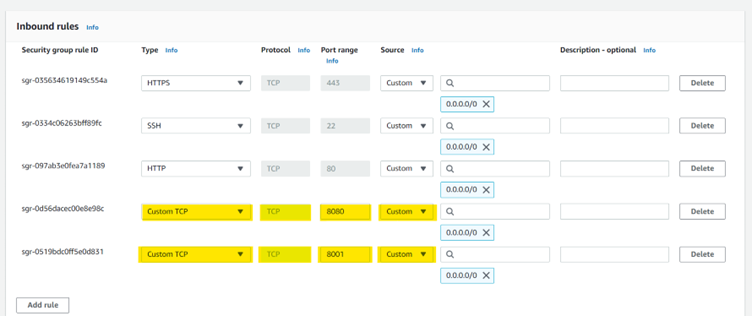
6. Install Docker as well to the machine by running,

“Sudo apt-install docker.io”

7. Now check if it got installed by running “jenkins — version” and “docker — version”

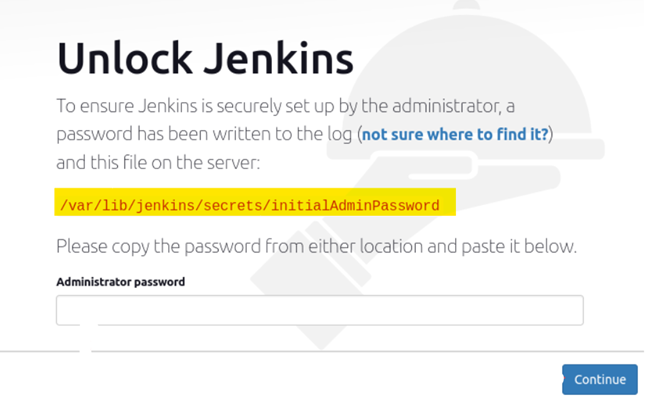


8. Now, we will allow ports 8080 and 8001 for this machine from a security group. We can find the security group in the VM description. Now, here we need to allow “Inbound Rule” as below:



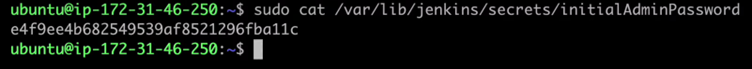
9. Now, Copy the Public Ip of the machine and paste it to the browser to access the Jenkins portal. As,

“54.193.49.139:8080”



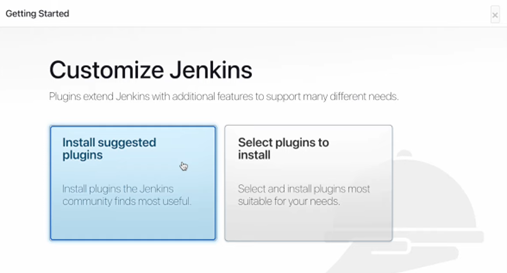
10. We need an Administrator Password to unlock this. For that, go to the provided highlighted path in the upper screenshot.

“cat /var/lib/Jenkins/secrets/initialAdminPassword”

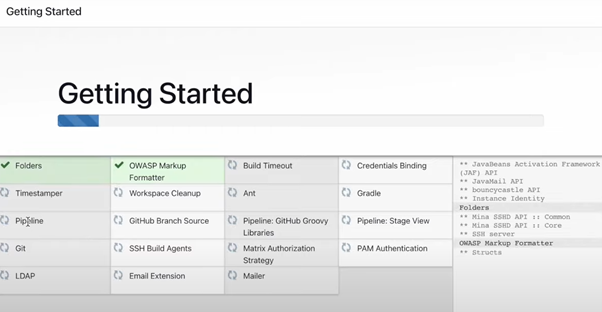


Paste this password in the “Administrator Password” Column.

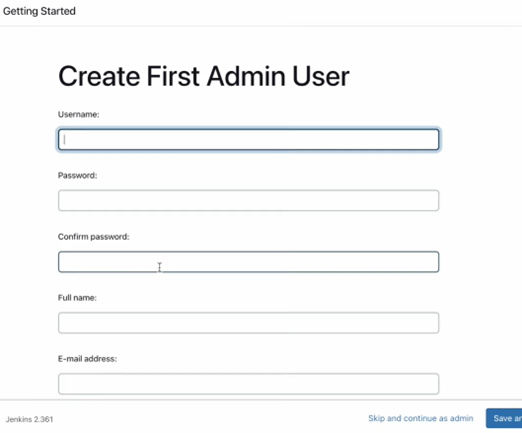
11. Now Click on, “Install Suggested Plugins”



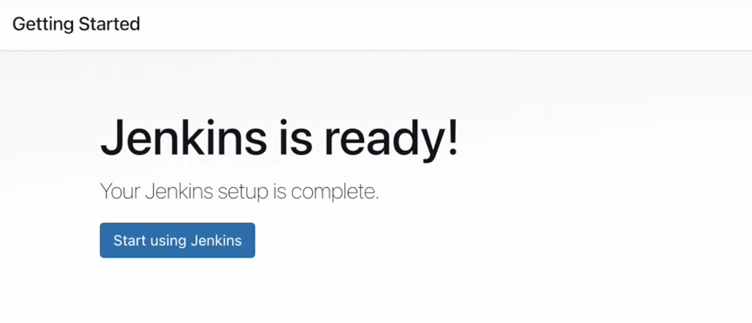
12. This will now install the suggested plugins. As



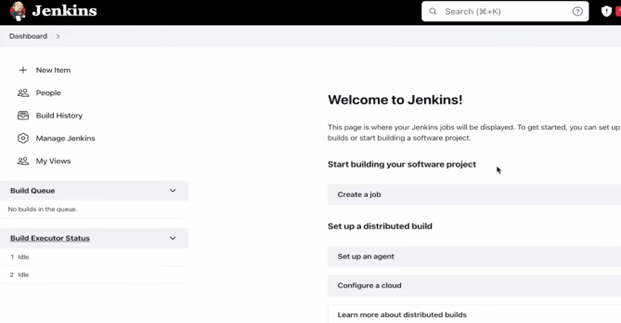
13. Now, Jenkins will ask us to create the First Admin User.



14. Add the fields accordingly.

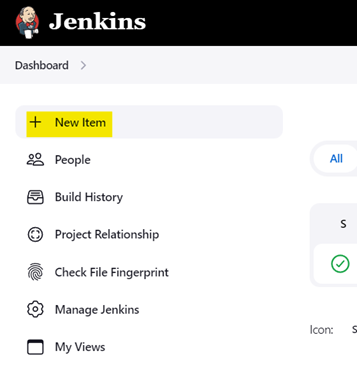


15. The Jenkins homepage will look like this,



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

17. From Jenkins Dashboard, Click on “New Item”.

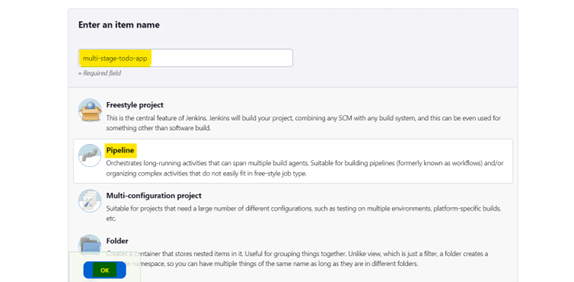


18. Now, Add the name as

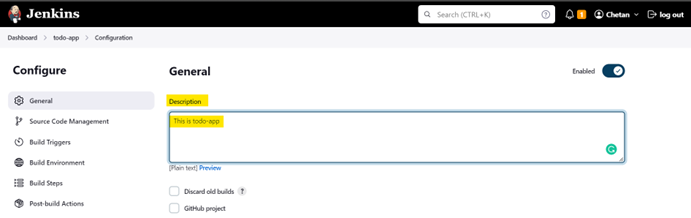
Name: todo-app

Project: Freestyle project

Click “Ok”.

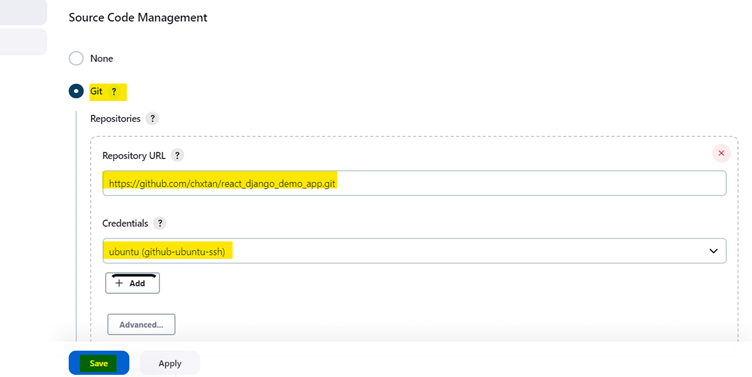


19. Here, we need to fill up the description.



20. In Source Code Management, select Git and Add Repository URL and Credentials.

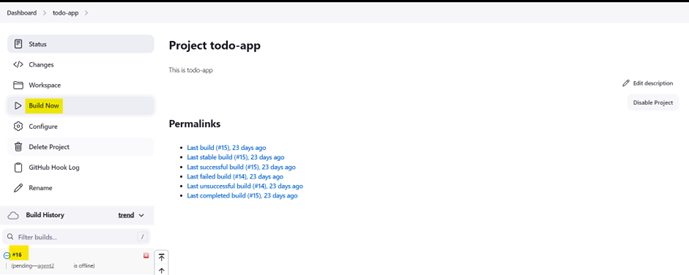
(If there is not any added credential, we need to add)



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



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

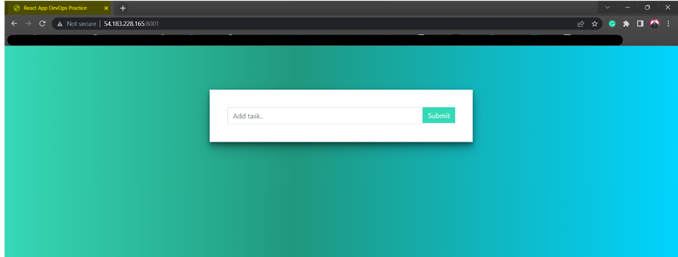


23. In Output Console,



24. After getting success, In the browser, search for

<public\_ip\_of\_ec2:8001>



Now, our goal is,

· Whenever the developer commits their code in GitHub, after every commit, it should reflect in the live web app.

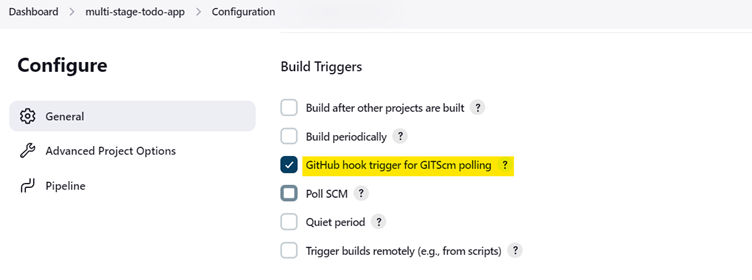
· For that, we will use “GitScm polling”.

· 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.

25. Now, configure the project again, and add

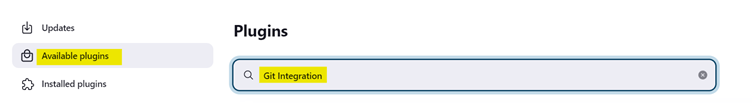
Build Trigger: GitHub hook trigger for GitScm polling.

Description: GitHub webhook integration



26. We need to install the “Git Integration” plugin from Manage Jenkins, by following the path,

(Manage Jenkins > Manage Plugins > Git Integration).

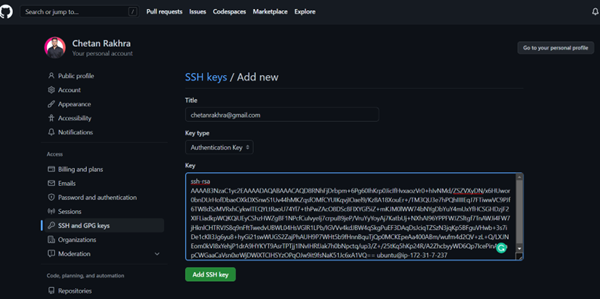


27. Now, Goto GitHub > Settings > SSH and GPG Keys > New SSH Key.

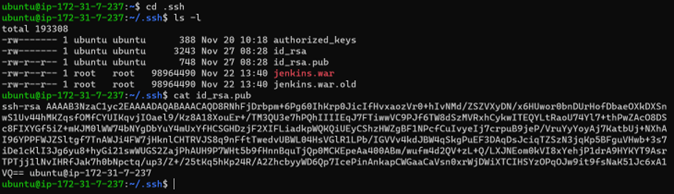
Add details as

Title: [chetanrakhra@gmail.com](http://mailto:chetanrakhra@gmail.com/)

Key type: <public\_key>



28. To get the Public key, open the “id\_rsa.pub” file and copy the content. (Public Key)



29. Now, we need to go to GitHub and create a new SSH and GPG Key.

GitHub > Repo “react-django-demo-app” > Settings > Webhooks.

30. Add the following details,

Payload URL: http://<public\_ip\_of\_ec2>:8080/github-webhook/

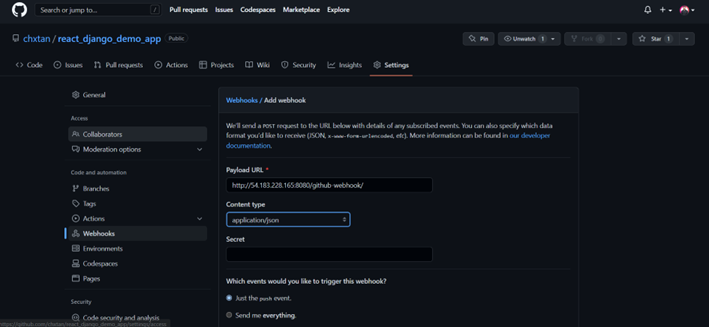
Content-Type: application/json

Which event would you like to trigger this webhook?

o Just the push event.

Active: True

Click on “Add Webhook”.



31. Now, Save the configured project.

32. Do some changes in the code and push to GitHub, this will automatically run a pipeline, and the new code will be Live.

— — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — —