



BRACT'S, Vishwakarma Institute of Information Technology, Pune -48

(An Autonomous Institute affiliated to Savitribai Phule Pune University)

(NBA and NAAC accredited, ISO 9001:2015 certified)

Department of Information Technology

Nodejs CI/CD Pipeline Using Jenkins and Docker with
GITHUB Integration

Roll No.	Name	GR No.
333003	Arshad Attar	22010173
333004	Ayush Gupta	22011191
333011	Vrushabh Chudiwal	22011174
333025	Kaustubh Joshi	22010030

INDEX

1. Technologies used
2. Steps
3. References

Technologies Used

Jenkins

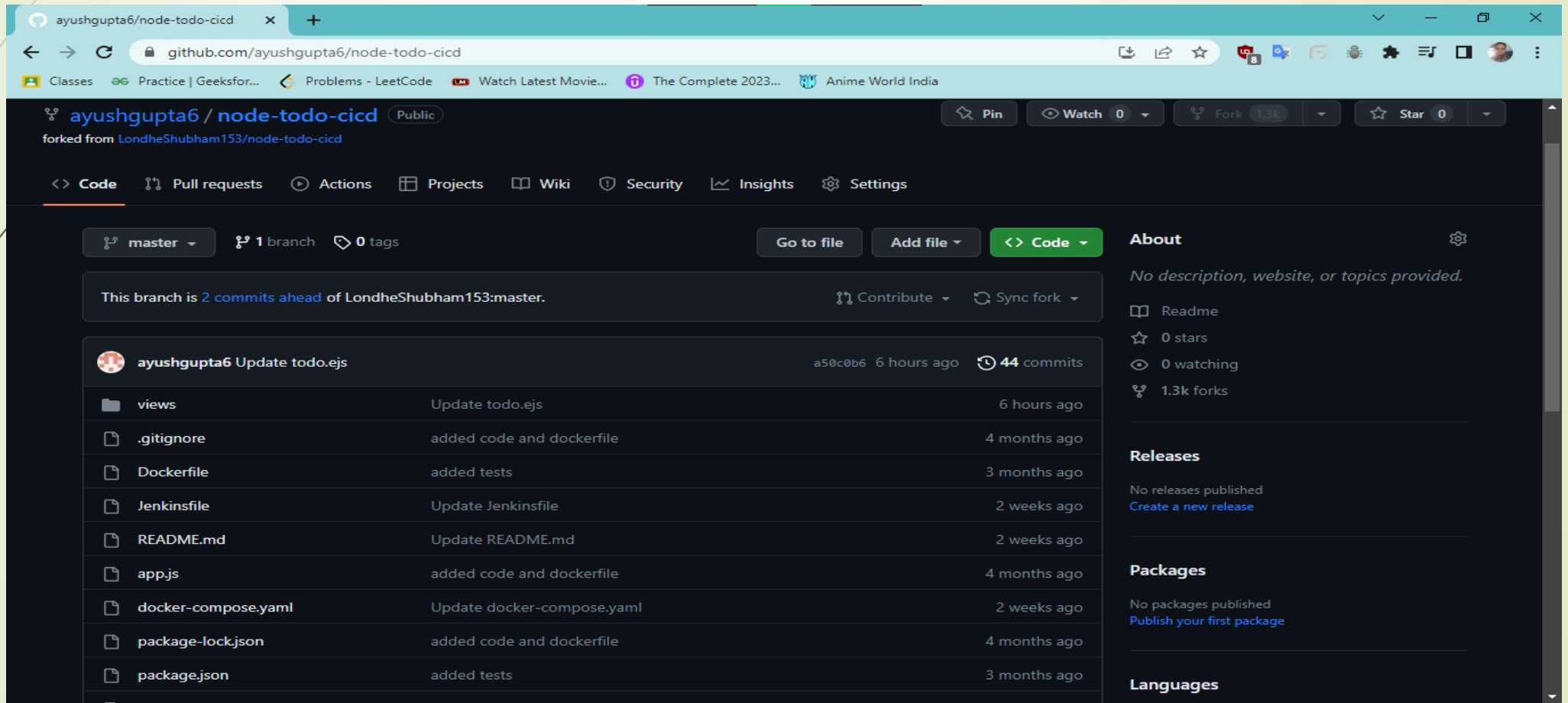
- Jenkins is a widely used open-source automation server that is used to build, test, and deploy software applications. Jenkins can be used to automate almost any task related to software development, including building and testing software, deploying applications, and managing infrastructure.
- Jenkins is highly extensible and customizable, with a vast library of over 1,500 plugins that allow developers to add new functionality and integrate with a wide range of tools and services. Jenkins plugins are available for popular tools and services such as Git, GitHub, Docker, AWS, and many others.
- Jenkins is designed to support continuous integration (CI) and continuous delivery (CD) workflows, allowing teams to automate the process of building, testing, and deploying software applications

Docker

- Docker is a popular platform that allows developers to package and deploy applications as lightweight, portable containers that can run virtually anywhere. With Docker, you can create containers that include everything your application needs to run, such as the code, runtime, system tools, libraries, and settings.
- Docker provides a flexible and efficient way to manage your application infrastructure, allowing you to easily scale up or down, roll out updates and changes, and manage dependencies across different environments. Docker's containerization technology also helps to reduce the risk of conflicts between applications, as each container is isolated from the host system and other containers.
- Docker has a large and active community of developers and users, who have contributed a wide range of tools, plugins, and integrations to extend its functionality. Some popular Docker-related tools and technologies include Docker Compose, Docker Swarm, Kubernetes, and Docker Hub, which is a repository for storing and sharing Docker images.

Step1: Forks the GIT Repo

Link: <https://github.com/ayushgupta6/node-todo-cicd.git>

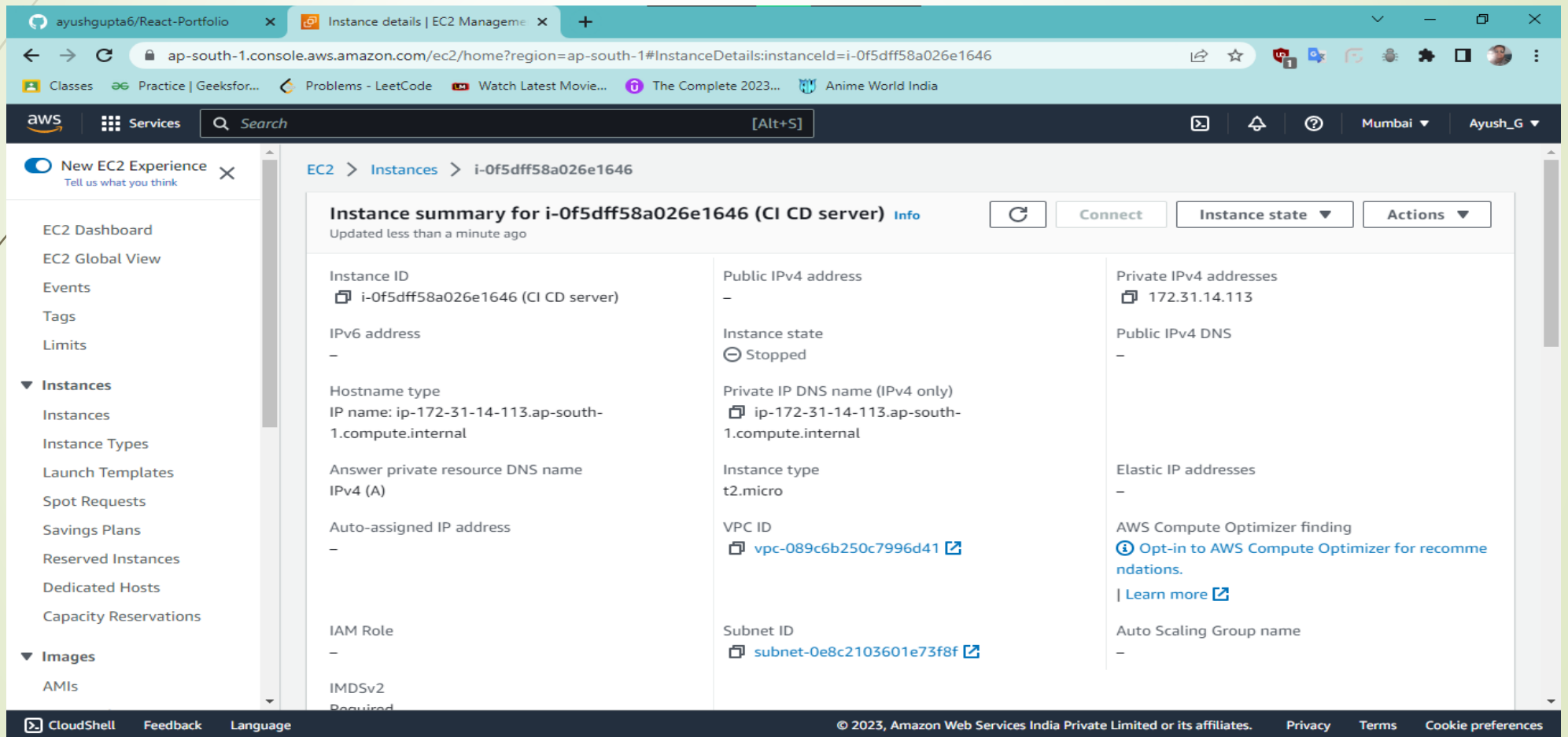


The screenshot shows a web browser displaying the GitHub repository page for 'ayushgupta6/node-todo-cicd'. The repository is public and was forked from 'LondheShubham153/node-todo-cicd'. The page includes navigation tabs for Code, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The 'Code' tab is active, showing the 'master' branch with 1 branch and 0 tags. A message indicates 'This branch is 2 commits ahead of LondheShubham153:master.' Below this, a list of files is shown with their commit history:

File	Commit Message	Time Ago
views	Update todo.ejs	6 hours ago
.gitignore	added code and dockerfile	4 months ago
Dockerfile	added tests	3 months ago
Jenkinsfile	Update Jenkinsfile	2 weeks ago
README.md	Update README.md	2 weeks ago
app.js	added code and dockerfile	4 months ago
docker-compose.yaml	Update docker-compose.yaml	2 weeks ago
package-lock.json	added code and dockerfile	4 months ago
package.json	added tests	3 months ago

On the right side, the 'About' section states 'No description, website, or topics provided.' and shows 0 stars, 0 watching, and 1.3k forks. The 'Releases' section indicates 'No releases published' with a link to 'Create a new release'. The 'Packages' section indicates 'No packages published' with a link to 'Publish your first package'. The 'Languages' section is partially visible at the bottom.

Step2: Create the EC-2 server on AWS



The screenshot displays the AWS Management Console interface for an EC2 instance. The browser address bar shows the URL: `ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#InstanceDetails:instanceId=i-0f5dff58a026e1646`. The console header includes the AWS logo, a search bar, and the user's name 'Ayush_G'.

The left sidebar contains navigation options: EC2 Dashboard, EC2 Global View, Events, Tags, Limits, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, and AMIs.

The main content area shows the 'Instance summary for i-0f5dff58a026e1646 (CI CD server)'. The instance is in a 'Stopped' state. Key details include:

- Instance ID:** i-0f5dff58a026e1646 (CI CD server)
- Public IPv4 address:** -
- Private IPv4 addresses:** 172.31.14.113
- Instance state:** Stopped
- Private IP DNS name (IPv4 only):** ip-172-31-14-113.ap-south-1.compute.internal
- Instance type:** t2.micro
- VPC ID:** vpc-089c6b250c7996d41
- Subnet ID:** subnet-0e8c2103601e73f8f
- IAM Role:** -
- IMDSv2:** Required

At the bottom of the console, there are links for CloudShell, Feedback, and Language, along with the copyright notice: © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy, Terms, and Cookie preferences are also available.

Security group Photo

The screenshot displays the AWS Management Console for a security group in the ap-south-1 region. The browser tabs include 'node-todo-cicd/Dockerfile at ma...', 'EC2 Management Console', and 'Sign in [Jenkins]'. The URL is 'ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#SecurityGroup:securityGroupId=sg-0cc3395d093585ef3'. The console header shows the AWS logo, 'Services', a search bar, and the user 'Ayush_G' in Mumbai.

On the left sidebar, the 'New EC2 Experience' toggle is on, and the navigation menu includes 'EC2 Dashboard', 'EC2 Global View', 'Events', 'Tags', 'Limits', 'Instances', 'Instance Types', 'Launch Templates', 'Spot Requests', 'Savings Plans', 'Reserved Instances', 'Dedicated Hosts', 'Capacity Reservations', 'Images', and 'AMIs'.

The main content area shows the security group details for 'sg-0cc3395d093585ef3'. The owner is '630230110523'. It has 4 inbound rules and 1 outbound rule. The 'Inbound rules' tab is selected, showing a notification about the Reachability Analyzer and a table of 4 inbound rules.

<input type="checkbox"/>	Name	Security group rule...	IP version	Type	Protocol	Port range
<input type="checkbox"/>	-	sgr-02b657b5d7268b...	IPv4	Custom TCP	TCP	8080
<input type="checkbox"/>	-	sgr-0d1a902daae10c279	IPv4	Custom TCP	TCP	8000
<input type="checkbox"/>	-	sgr-036f14a0e80b649c9	IPv4	Custom TCP	TCP	8003
<input type="checkbox"/>	-	sgr-0bb0bdaa0ef6162cc	IPv4	SSH	TCP	22

The footer contains links for 'CloudShell', 'Feedback', 'Language', and copyright information for Amazon Web Services India Private Limited or its affiliates, along with 'Privacy', 'Terms', and 'Cookie preferences' links.

Step4.1: Jenkins Install

```
root@ip-172-31-14-113:/home/ec2-user
[root@ip-172-31-14-113 ec2-user]# wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo
--2023-04-27 09:33:50-- https://pkg.jenkins.io/redhat-stable/jenkins.repo
Resolving pkg.jenkins.io (pkg.jenkins.io)... 151.101.154.133, 2a04:4e42:24::645
Connecting to pkg.jenkins.io (pkg.jenkins.io)|151.101.154.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 85
Saving to: '/etc/yum.repos.d/jenkins.repo'

/etc/yum.repos.d/jenkins. 100%[=====>]          85  --.-KB/s    in 0s

2023-04-27 09:33:51 (5.63 MB/s) - '/etc/yum.repos.d/jenkins.repo' saved [85/85]

[root@ip-172-31-14-113 ec2-user]# rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
[root@ip-172-31-14-113 ec2-user]# yum upgrade
Jenkins-stable                               20 kB/s | 26 kB    00:01
Dependencies resolved.
Nothing to do.
Complete!
[root@ip-172-31-14-113 ec2-user]#
```


Step4.2: Jenkins Install

```

root@ip-172-31-14-113:/home/ec2-user
[root@ip-172-31-14-113 ec2-user]# yum install jenkins -y
Last metadata expiration check: 0:04:55 ago on Thu Apr 27 09:34:33 2023.
Dependencies resolved.
=====
Package                Architecture      Version           Repository        Size
=====
Installing:
jenkins                noarch            2.387.2-1.1      jenkins           94 M
=====
Transaction Summary
=====
Install 1 Package

Total download size: 94 M
Installed size: 94 M
Downloading Packages:
jenkins-2.387.2-1.1.noarch.rpm                    5.6 MB/s | 94 MB  00:16
-----
Total                                              5.6 MB/s | 94 MB  00:16
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing      : 1/1
  Running scriptlet: jenkins-2.387.2-1.1.noarch 1/1
  Installing      : jenkins-2.387.2-1.1.noarch 1/1
  Running scriptlet: jenkins-2.387.2-1.1.noarch 1/1
  Verifying       : jenkins-2.387.2-1.1.noarch 1/1

Installed:
jenkins-2.387.2-1.1.noarch

Complete!
[root@ip-172-31-14-113 ec2-user]#

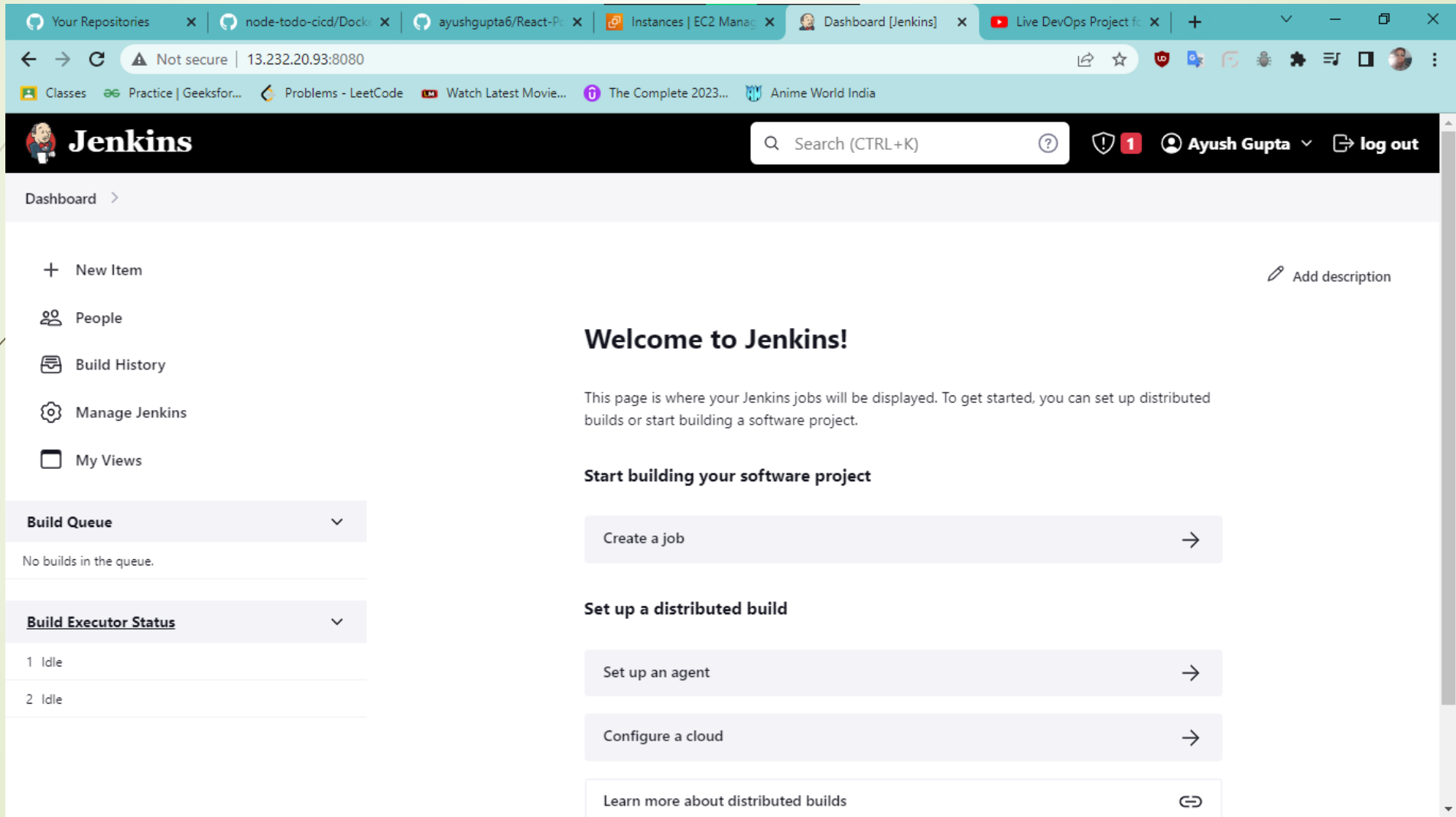
```

Step4.3: Jenkins Install

```
root@ip-172-31-14-113:/home/ec2-user
[root@ip-172-31-14-113 ec2-user]# systemctl enable jenkins
Synchronizing state of jenkins.service with SysV service script with /usr/lib/systemd/systemd-sysv-inst
all.
Executing: /usr/lib/systemd/systemd-sysv-install enable jenkins
Created symlink /etc/systemd/system/multi-user.target.wants/jenkins.service → /usr/lib/systemd/system/j
enkins.service.
[root@ip-172-31-14-113 ec2-user]# systemctl start jenkins
^C
[root@ip-172-31-14-113 ec2-user]# systemctl status jenkins
• jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: disabled)
   Active: activating (start) since Thu 2023-04-27 09:41:20 UTC; 20s ago
   Main PID: 26652 (java)
   Tasks: 41 (limit: 1112)
   Memory: 311.8M
   CPU: 18.003s
   CGroup: /system.slice/jenkins.service
           └─26652 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot>

Apr 27 09:41:31 ip-172-31-14-113.ap-south-1.compute.internal jenkins[26652]: *****>
Apr 27 09:41:31 ip-172-31-14-113.ap-south-1.compute.internal jenkins[26652]: *****>
Apr 27 09:41:31 ip-172-31-14-113.ap-south-1.compute.internal jenkins[26652]: *****>
Apr 27 09:41:31 ip-172-31-14-113.ap-south-1.compute.internal jenkins[26652]: Jenkins initial setup is >
Apr 27 09:41:31 ip-172-31-14-113.ap-south-1.compute.internal jenkins[26652]: Please use the following >
Apr 27 09:41:31 ip-172-31-14-113.ap-south-1.compute.internal jenkins[26652]: a1962593f3174ec19684c12f6>
Apr 27 09:41:31 ip-172-31-14-113.ap-south-1.compute.internal jenkins[26652]: This may also be found at>
Apr 27 09:41:31 ip-172-31-14-113.ap-south-1.compute.internal jenkins[26652]: *****>
Apr 27 09:41:31 ip-172-31-14-113.ap-south-1.compute.internal jenkins[26652]: *****>
Apr 27 09:41:31 ip-172-31-14-113.ap-south-1.compute.internal jenkins[26652]: *****>
lines 1-20/20 (END)
```

Step4.4: Jenkins Install



The screenshot shows the Jenkins web interface in a browser. The browser's address bar displays '13.232.20.93:8080'. The Jenkins dashboard includes a sidebar with navigation links: 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'My Views'. The main content area features a 'Welcome to Jenkins!' message, a 'Start building your software project' section with a 'Create a job' button, and a 'Set up a distributed build' section with buttons for 'Set up an agent', 'Configure a cloud', and a link to 'Learn more about distributed builds'. On the left, the 'Build Queue' is empty, and the 'Build Executor Status' shows two idle executors.

Jenkins

Search (CTRL+K)

Dashboard

- + New Item
- People
- Build History
- Manage Jenkins
- My Views

Build Queue

No builds in the queue.

Build Executor Status

- 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.

Start building your software project

Create a job

Set up a distributed build

Set up an agent

Configure a cloud

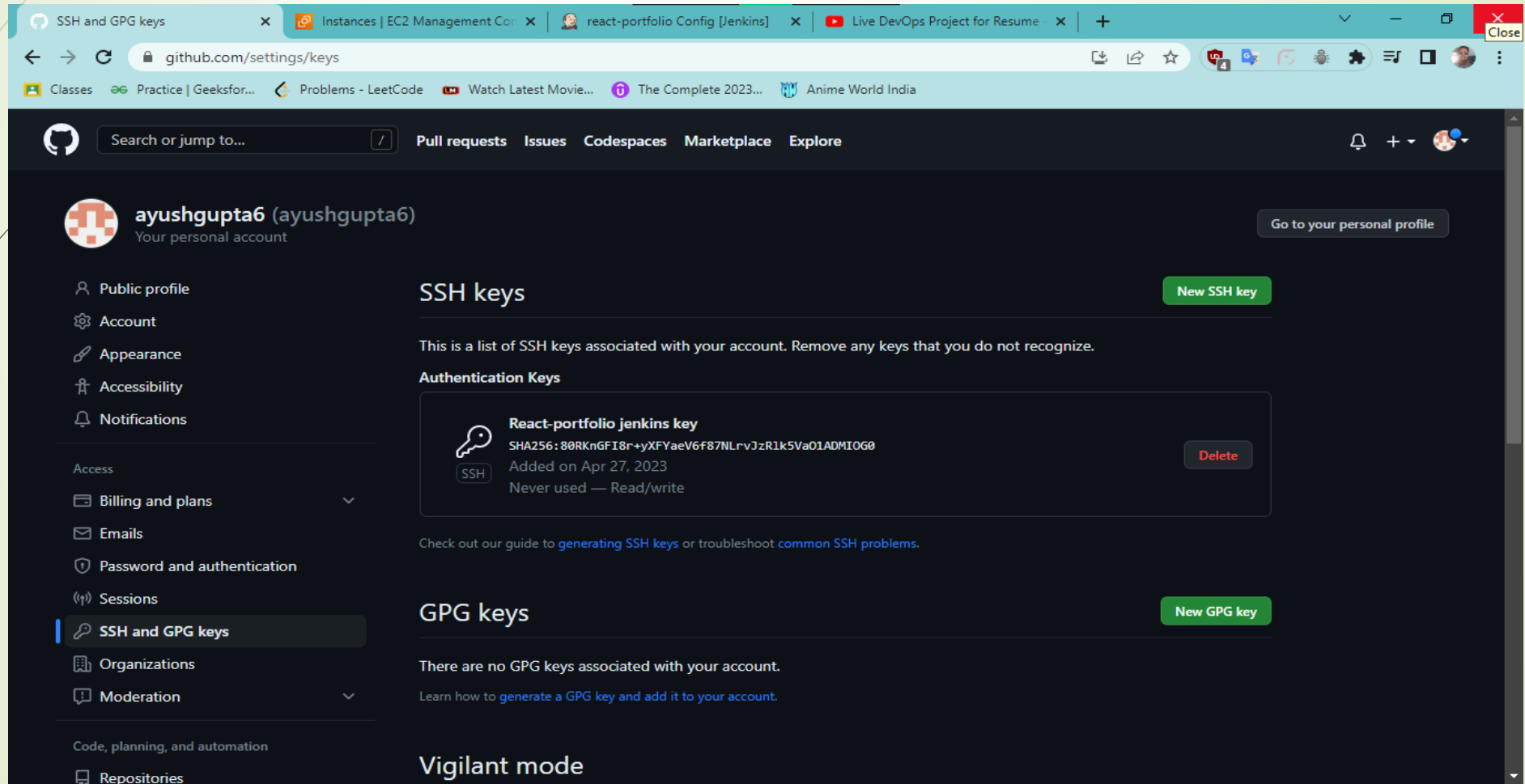
Learn more about distributed builds

Step5.1: Key Generation and Adding

```
root@ip-172-31-14-113:/home/ec2-user
[ec2-user@ip-172-31-14-113 ~]$ sudo su
[root@ip-172-31-14-113 ec2-user]# ls /var/lib/jenkins/secrets/initialAdminPassword
/var/lib/jenkins/secrets/initialAdminPassword
[root@ip-172-31-14-113 ec2-user]# cat /var/lib/jenkins/secrets/initialAdminPassword
a1962593f3174ec19684c12f60012d87
[root@ip-172-31-14-113 ec2-user]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:80RKnGFI8r+yXFYaeV6f87NLRvJzRlk5VaO1ADMIOG0 root@ip-172-31-14-113.ap-south-1.compute.internal
The key's randomart image is:
+---[RSA 3072]---+
|  ..+o+ .+.. oo|
|  =.E +  o + +|
|  + + .  . .o|
|  o +    o.|
|  S + .  +|
|  @ . . + |
|  . = o  =. |
|  . =    .o* |
|  o      oo+=|
+-----[SHA256]-----+
[root@ip-172-31-14-113 ec2-user]#
```

Step5.2: Key Generation and Adding

Adding Public SSH to the Github



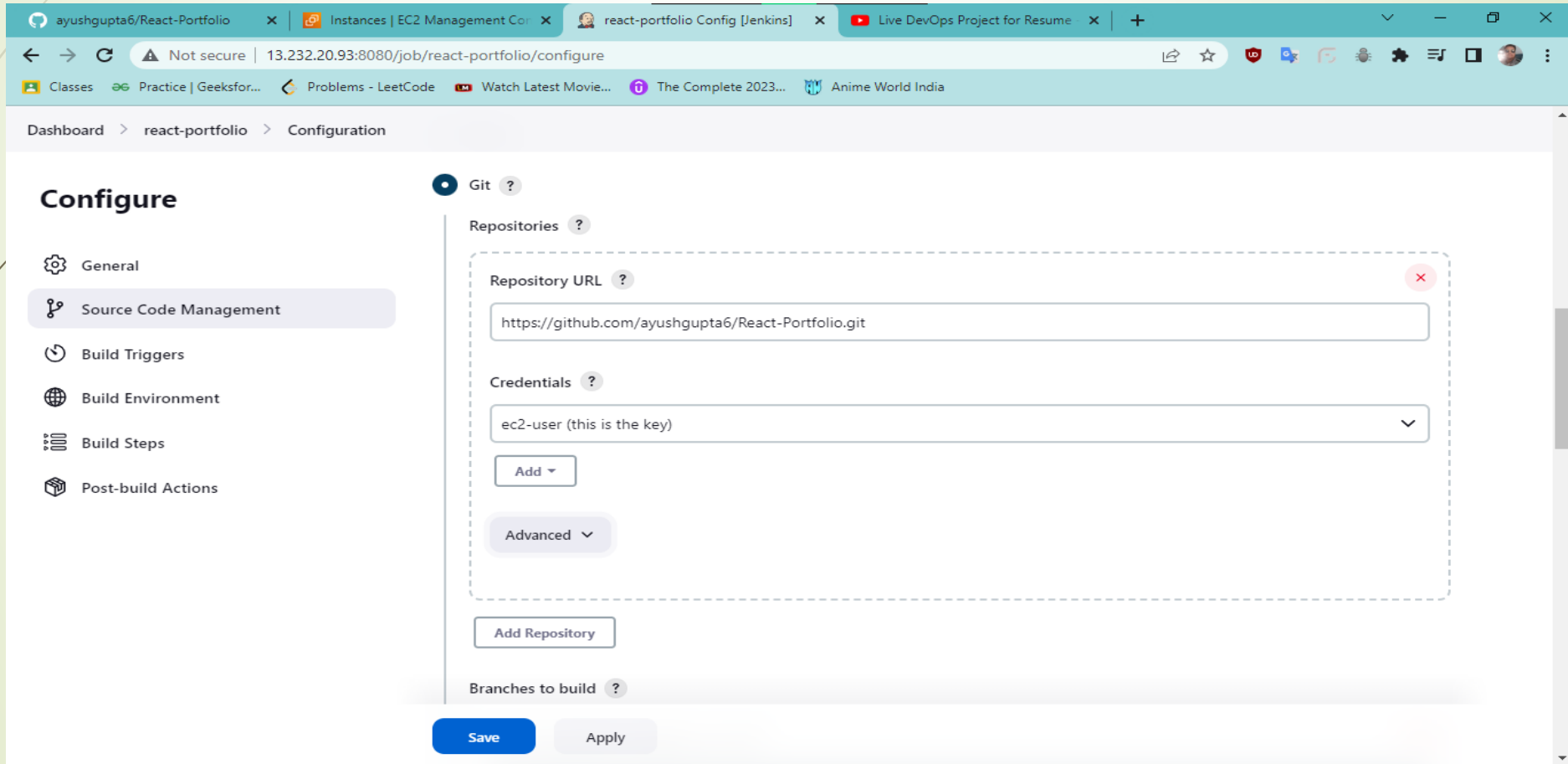
The screenshot shows the GitHub 'SSH and GPG keys' settings page for the user 'ayushgupta6'. The page is divided into three main sections: SSH keys, GPG keys, and Vigilant mode.

- SSH keys:** This section contains a list of SSH keys associated with the account. One key is listed: 'React-portfolio jenkins key' with a SHA256 hash of 80RKnGFI8r+yXFYaeV6f87NLrVJzR1k5Va01ADMI0G0. It was added on April 27, 2023, and has never been used. The permissions are 'Read/write'. A 'Delete' button is visible next to the key.
- GPG keys:** This section states that there are no GPG keys associated with the account. It includes a link to learn how to generate a GPG key and add it to the account.
- Vigilant mode:** This section is currently empty.

The left sidebar shows the user's profile and navigation links, with 'SSH and GPG keys' selected. The top navigation bar includes links for Pull requests, Issues, Codespaces, Marketplace, and Explore.

Step5.3: Key Generation and Adding

Adding private key to the Jenkins



The screenshot shows the Jenkins web interface in a browser. The address bar indicates the URL is `13.232.20.93:8080/job/react-portfolio/configure`. The page title is "Configure" and the breadcrumb trail is "Dashboard > react-portfolio > Configuration". On the left sidebar, the "Source Code Management" option is selected. The main content area is titled "Git" and contains a "Repositories" section. Inside this section, a "Repository URL" field is populated with `https://github.com/ayushgupta6/React-Portfolio.git`. Below it, a "Credentials" dropdown menu is set to "ec2-user (this is the key)". There are "Add" and "Advanced" buttons below the credentials field. At the bottom of the repository configuration area is an "Add Repository" button. Below the repository list is a "Branches to build" section. At the very bottom of the configuration page are "Save" and "Apply" buttons.

ayushgupta6/React-Portfolio x Instances | EC2 Management Co... react-portfolio Config [Jenkins] x Live DevOps Project for Resume x +

Not secure | 13.232.20.93:8080/job/react-portfolio/configure

Classes Practice | Geeksfor... Problems - LeetCode Watch Latest Movie... The Complete 2023... Anime World India

Dashboard > react-portfolio > Configuration

Configure

- General
- Source Code Management
- Build Triggers
- Build Environment
- Build Steps
- Post-build Actions

Git ?

Repositories ?

Repository URL ?

`https://github.com/ayushgupta6/React-Portfolio.git`

Credentials ?

ec2-user (this is the key)

Add

Advanced

Add Repository

Branches to build ?

Save Apply

DockerFile Code

```
FROM node:12.2.0-alpine
```

```
WORKDIR app
```

```
COPY . .
```

```
RUN npm install
```

```
RUN npm run test
```

```
EXPOSE 8000
```

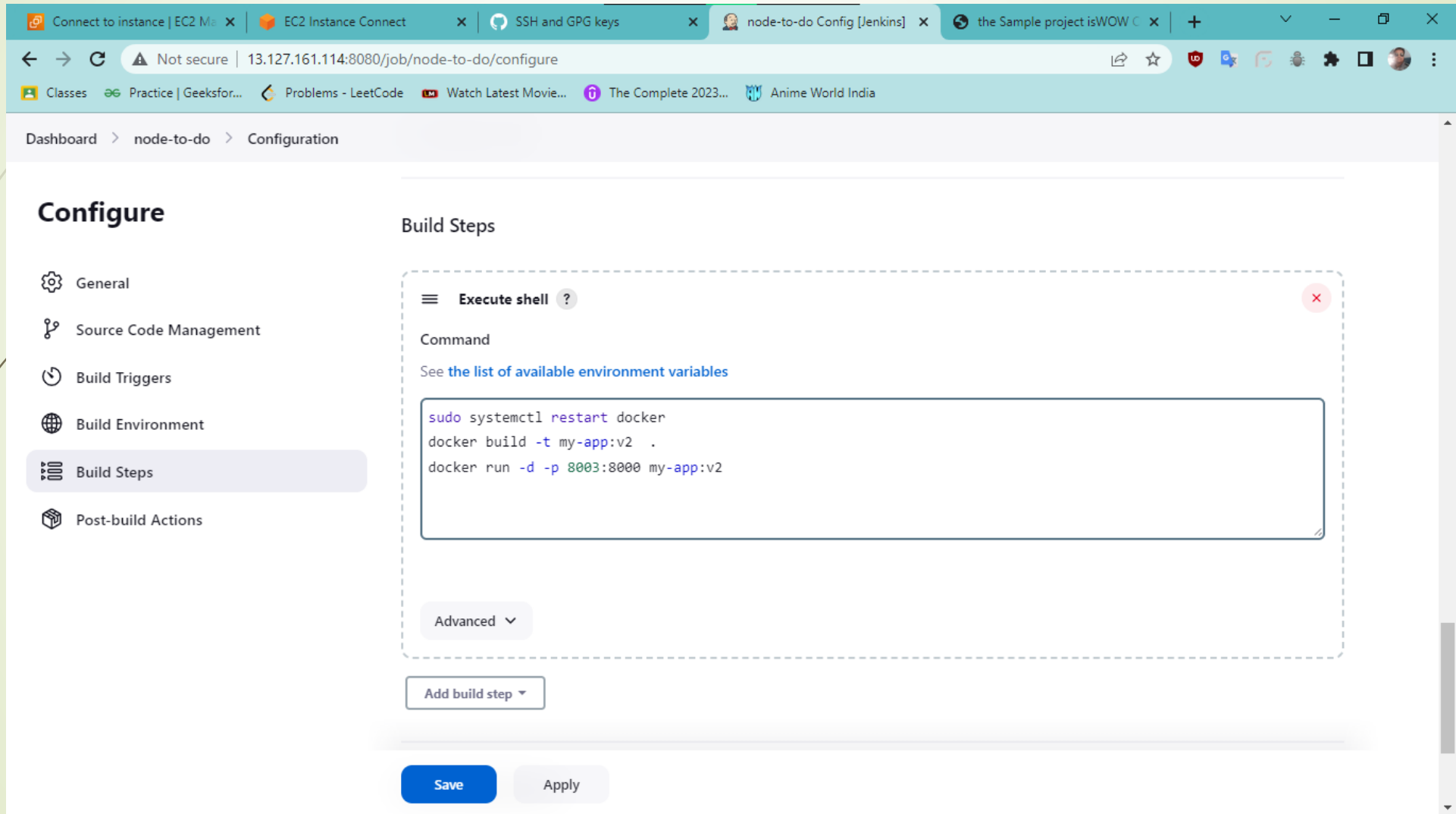
```
CMD ["node","app.js"]
```

Step 6: Docker Installing and Running

```
ec2-user@ip-172-31-35-213:/var/lib/jenkins/workspace/node-to-do
[ec2-user@ip-172-31-35-213 node-to-do]$ sudo docker build -t my-app:v1 .
Sending build context to Docker daemon 25.23MB
Step 1/7 : FROM node:12.2.0-alpine
--> f391dabf9dce
Step 2/7 : WORKDIR app
--> Using cache
--> 90f582c3d0e7
Step 3/7 : COPY . .
--> Using cache
--> 806be67cbbae
Step 4/7 : RUN npm install
--> Using cache
--> afdadc039d8b
Step 5/7 : RUN npm run test
--> Using cache
--> 50312806314d
Step 6/7 : EXPOSE 8000
--> Using cache
--> 59509a4e69ec
Step 7/7 : CMD ["node","app.js"]
--> Using cache
--> c5d821f90037
Successfully built c5d821f90037
Successfully tagged my-app:v1
[ec2-user@ip-172-31-35-213 node-to-do]$
```

```
ec2-user@ip-172-31-35-213:/var/lib/jenkins/workspace/node-to-do
[ec2-user@ip-172-31-35-213 node-to-do]$ sudo docker run -d -p 8005:8000 my-app:v1
194abde2a4e741f012939943cd421ce5421ee504d670ddac755f6cddca787029
[ec2-user@ip-172-31-35-213 node-to-do]$ docker ps
Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get "http://%2Fvar%2Frun%2Fdocker.sock/v1.24/containers/json": dial unix /var/run/docker.sock: connect: permission denied
[ec2-user@ip-172-31-35-213 node-to-do]$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                               NAMES
194abde2a4e7   my-app:v1 "node app.js"           16 seconds ago Up 16 seconds  0.0.0.0:8005->8000/tcp, :::8005->8000/tcp  friendly_mahavira
14865d743259   my-app:v2 "node app.js"           11 minutes ago Up 11 minutes  0.0.0.0:8003->8000/tcp, :::8003->8000/tcp  compassionate_taussig
[ec2-user@ip-172-31-35-213 node-to-do]$
```

Step7: Executable Shell



The screenshot shows the Jenkins web interface for configuring a job named 'node-to-do'. The browser tabs include 'Connect to instance | EC2 M...', 'EC2 Instance Connect', 'SSH and GPG keys', 'node-to-do Config [Jenkins]', and 'the Sample project isWOW'. The address bar shows '13.127.161.114:8080/job/node-to-do/configure'. The breadcrumb trail is 'Dashboard > node-to-do > Configuration'.

Configure

- General
- Source Code Management
- Build Triggers
- Build Environment
- Build Steps**
- Post-build Actions

Build Steps

Execute shell ?

Command

See [the list of available environment variables](#)

```
sudo systemctl restart docker
docker build -t my-app:v2 .
docker run -d -p 8003:8000 my-app:v2
```

Advanced ▾

Add build step ▾

Save Apply

Step8: Console Output

node-to-do/8/console

Watch Latest Movie... The Complete 2023... Anime World India

Console Output

```

Started by user Ayush
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/node-to-do
The recommended git tool is: NONE
using credential jenkins-sample
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/node-to-do/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/ayushgupta6/node-todo-cicd.git # timeout=10
Fetching upstream changes from https://github.com/ayushgupta6/node-todo-cicd.git
> git --version # timeout=10
> git --version # 'git version 2.39.2'
using GIT_SSH to set credentials this for jenkins project
Verifying host key using known hosts file
You're using 'Known hosts file' strategy to verify ssh host keys, but your known_hosts file does not exist, please go to
'Manage Jenkins' -> 'Configure Global Security' -> 'Git Host Key Verification Configuration' and configure host key
verification.
> git fetch --tags --force --progress -- https://github.com/ayushgupta6/node-todo-cicd.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision a50c0b690c58a97ea121f0f2668a08c7ea431431 (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f a50c0b690c58a97ea121f0f2668a08c7ea431431 # timeout=10
Commit message: "Update todo.ejs"

```

Sending build context to Docker daemon 25.23MB

Step 1/7 : FROM node:12.2.0-alpine

---> f391dabf9dce

Step 2/7 : WORKDIR app

---> Using cache

---> 90f582c3d0e7

Step 3/7 : COPY . .

---> Using cache

---> 806be67cbbae

Step 4/7 : RUN npm install

---> Using cache

---> afdadc039d8b

Step 5/7 : RUN npm run test

---> Using cache

---> 50312806314d

Step 6/7 : EXPOSE 8000

---> Using cache

---> 59509a4e69ec

Step 7/7 : CMD ["node","app.js"]

---> Using cache

---> c5d821f90037

Successfully built c5d821f90037

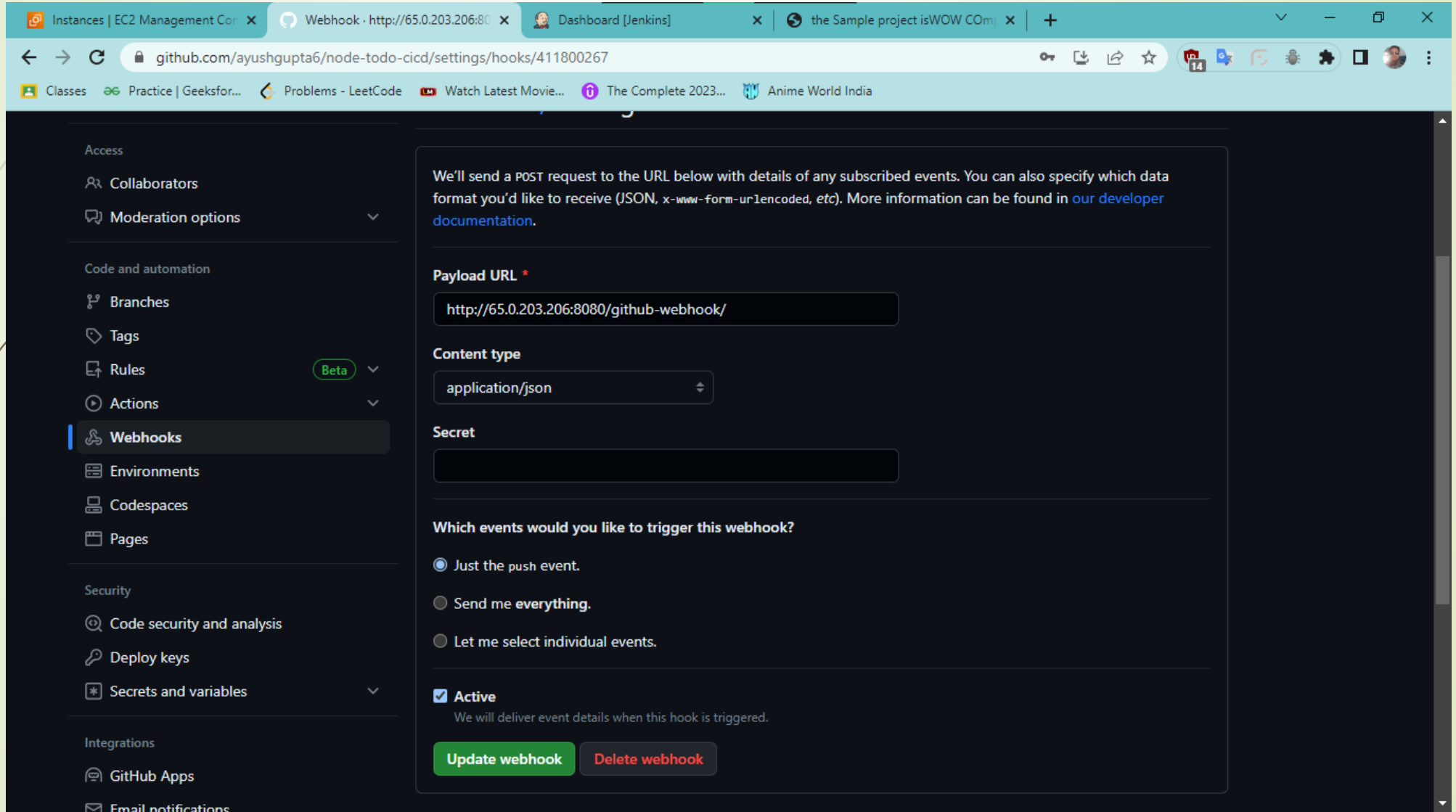
Successfully tagged my-app:v2

+ docker run -d -p 8003:8000 my-app:v2

14865d743259ad484ddb2a56907b65b023e099049d277e3252ade79755b5310c

Finished: SUCCESS

Step9: GitHub integration(WebHooks)



The screenshot shows the GitHub Webhook settings page for a repository. The browser tabs include 'Instances | EC2 Management Console', 'Webhook · http://65.0.203.206:8080', 'Dashboard [Jenkins]', and 'the Sample project isWOW COM'. The address bar shows the URL 'github.com/ayushgupta6/node-todo-cicd/settings/hooks/411800267'. The left sidebar contains navigation links: 'Access' (Collaborators, Moderation options), 'Code and automation' (Branches, Tags, Rules, Actions, Webhooks, Environments, Codespaces, Pages), 'Security' (Code security and analysis, Deploy keys, Secrets and variables), and 'Integrations' (GitHub Apps, Email notifications). The main content area is titled 'Webhook · http://65.0.203.206:8080' and contains the following information:

- Access:** Collaborators, Moderation options
- Code and automation:** Branches, Tags, Rules (Beta), Actions, **Webhooks**, Environments, Codespaces, Pages
- Security:** Code security and analysis, Deploy keys, Secrets and variables
- Integrations:** GitHub Apps, Email notifications

The main content area displays the following information:

- Access:** Collaborators, Moderation options
- Code and automation:** Branches, Tags, Rules (Beta), Actions, **Webhooks**, Environments, Codespaces, Pages
- Security:** Code security and analysis, Deploy keys, Secrets and variables
- Integrations:** GitHub Apps, Email notifications

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://65.0.203.206:8080/github-webhook/

Content type

application/json

Secret

Which events would you like to trigger this webhook?

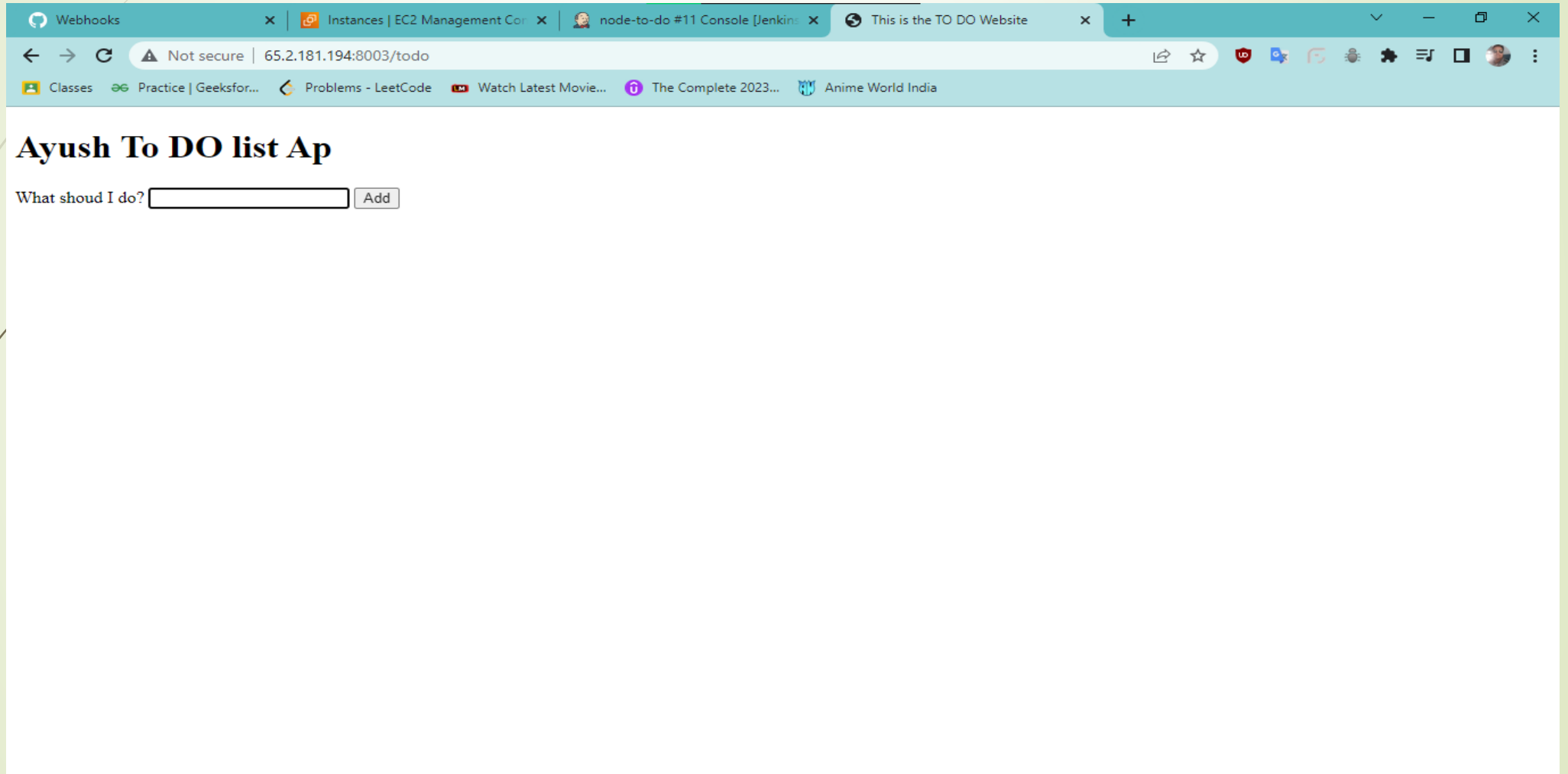
- ☒ Just the push event.
- ☐ Send me everything.
- ☐ Let me select individual events.

☒ **Active**

We will deliver event details when this hook is triggered.

Update webhook **Delete webhook**

Step10: WebSite



References

- <https://www.youtube.com/live/nplH3BzKHPk?feature=share>
- https://youtu.be/l_FmJGtOePk
- <https://www.jenkins.io/doc/>

THANK YOU