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

(An Autonomous Institute affiliated to Savitribai Phule Pune University)

(NBA and NAAC accredited$_1$, 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

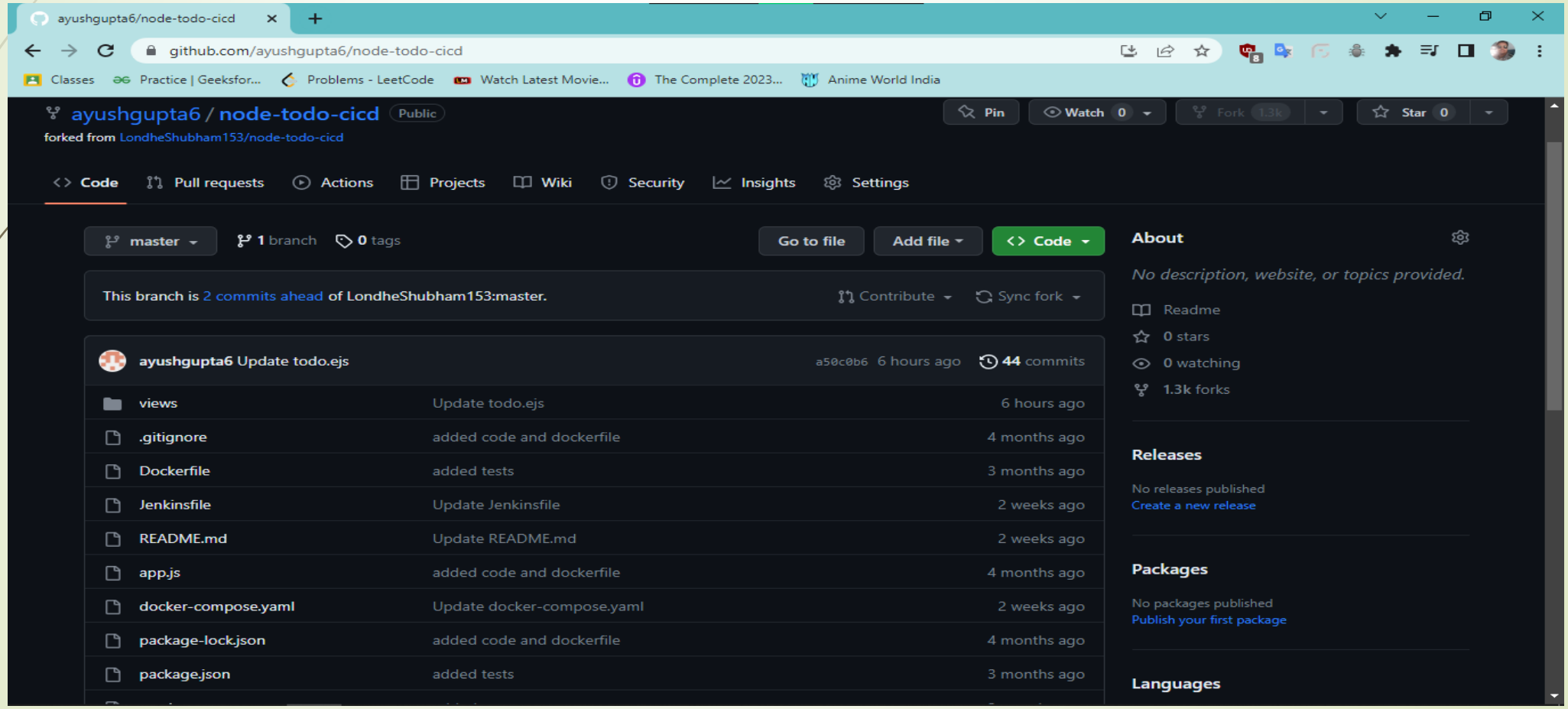
- Jenkins is a widely used open-source automation server that is used to build, test, and deploy software applications. Jenkins can be used to automate almost any task related to software development, including building and testing software, deploying applications, and managing infrastructure.
- Jenkins 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

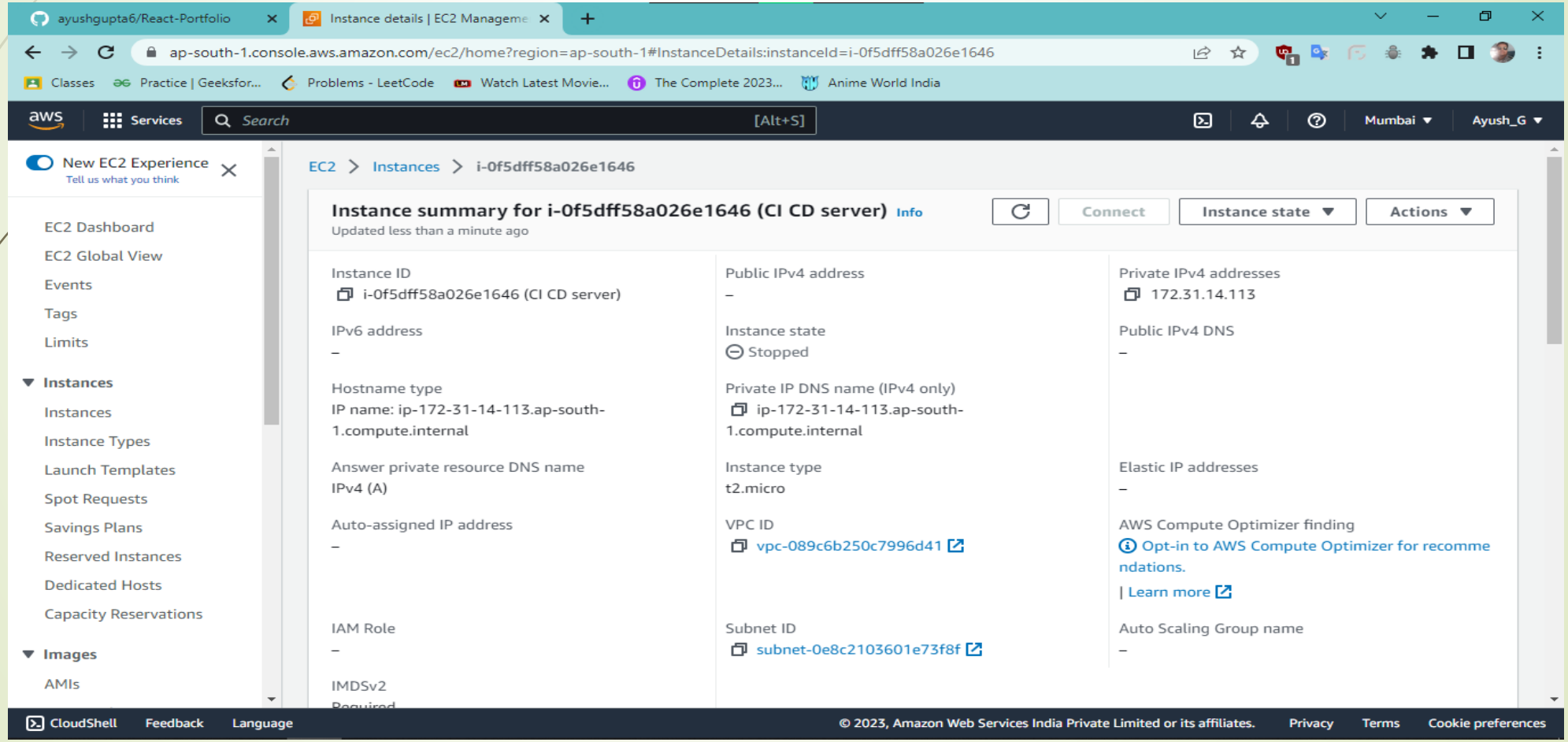
- Docker is a popular platform that allows developers to package and deploy applications as lightweight, portable containers that can run virtually anywhere. With Docker, you can create containers that include everything your application needs to run, such as the code, runtime, system tools, libraries, 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

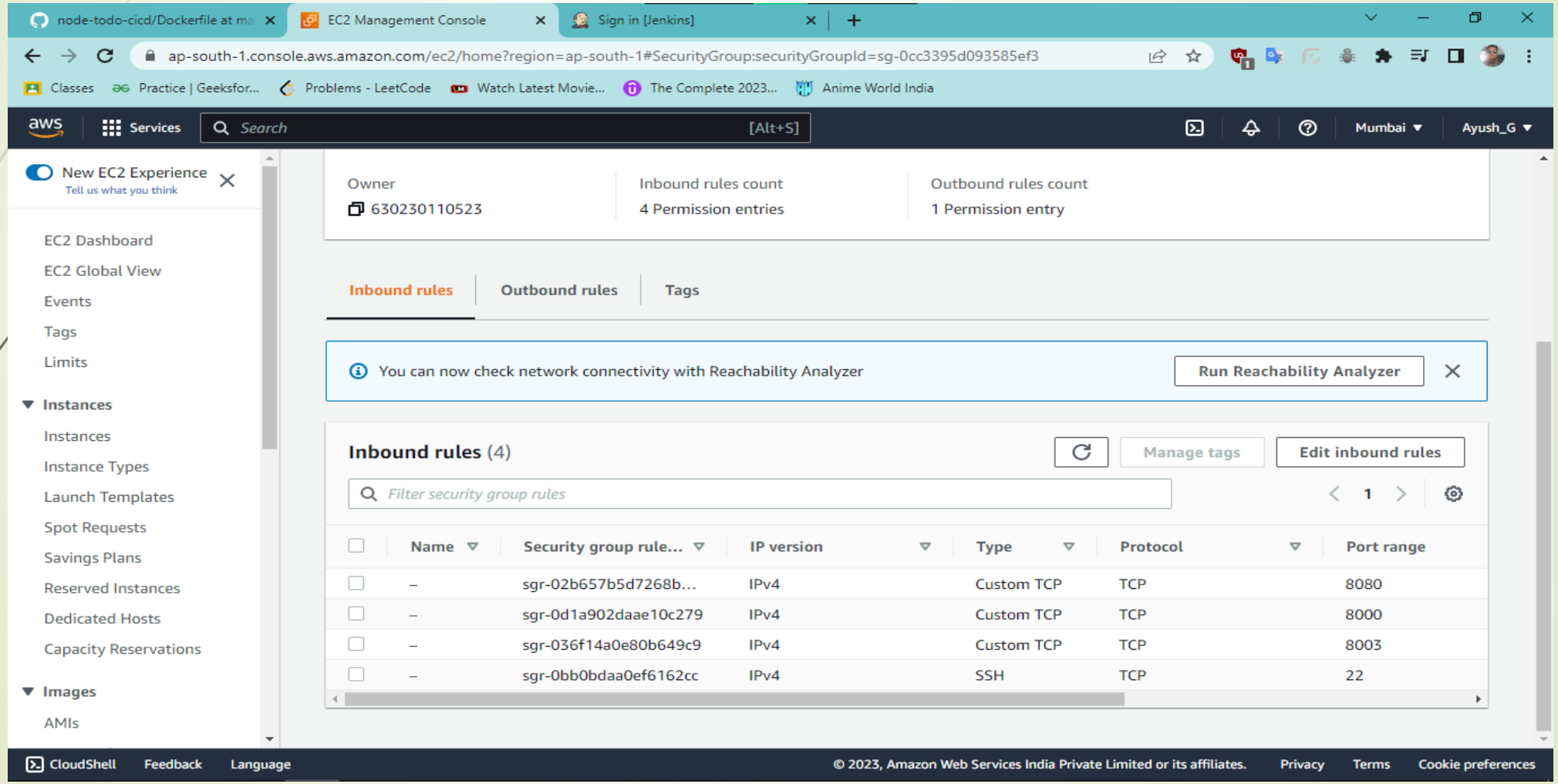
Link: https://github.com/ayushgupta6/node-todo-cicd.git



Step2: Create the EC-2 server on AWS



Security group Photo



Step3: Connect to our EC-2 server

We connected through SSH

```
root@ip-172-31-14-113:/home/ec2-user
PS C:\Users\jwosk\OneDrive\Desktop\College\SEM6\CC> ssh -i "ayush-key.pem" ec2-user@ec2-13-232-20-93.ap
-south-1.compute.amazonaws.com
The authenticity of host 'ec2-13-232-20-93.ap-south-1.compute.amazonaws.com (13.232.20.93)' can't be es
tablished.
ECDSA key fingerprint is SHA256:XdCiiMHGQE4r7+dAsMoozAzsij7wZm8Va5QWBT/krOY.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-13-232-20-93.ap-south-1.compute.amazonaws.com,13.232.20.93' (ECDSA) to
the list of known hosts.
       #_
       ####_          Amazon Linux 2023
      \_#####\
          \###|
            \#/ ___    https://aws.amazon.com/linux/amazon-linux-2023
             V~' '->
              /
            _/
          _/ _/
        _/m/'
Last login: Sun Apr 23 11:22:23 2023 from 13.233.177.3
[ec2-user@ip-172-31-14-113 ~]$ sudo su
[root@ip-172-31-14-113 ec2-user]#
```

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/redh
at-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]#
```

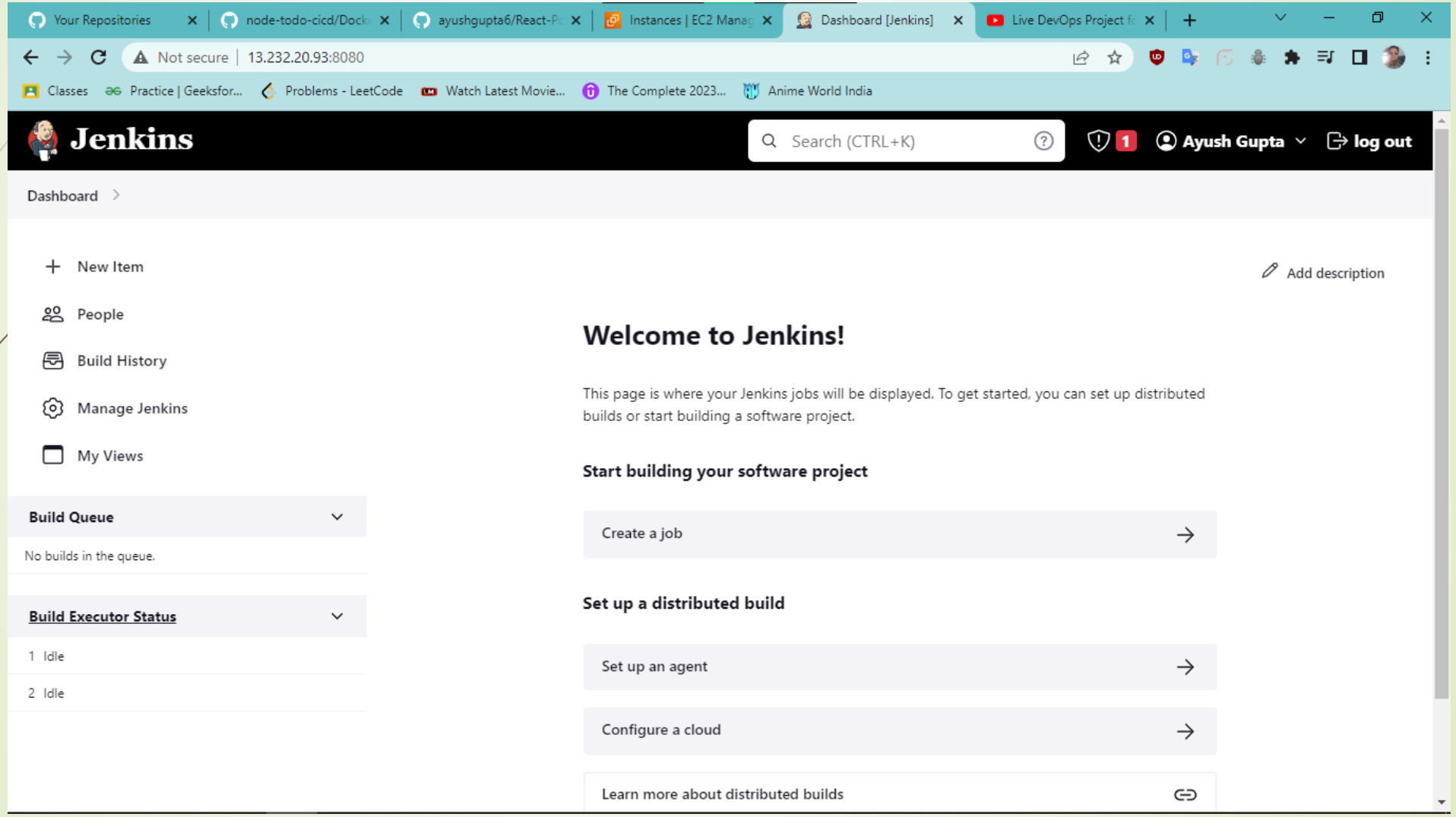
Step4.2: Jenkins Install

```
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



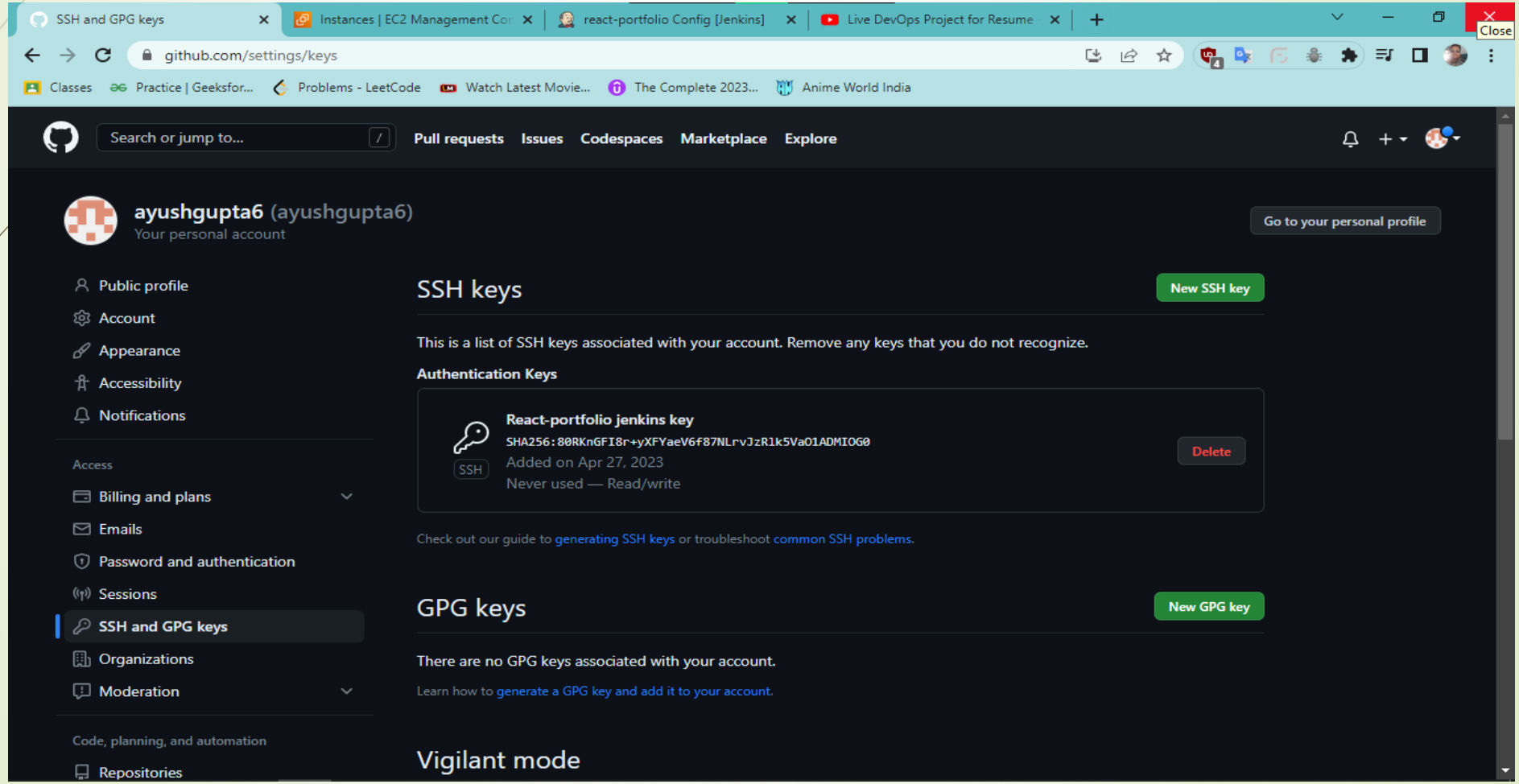
Step5.1: Key Generation and Adding

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]#
```

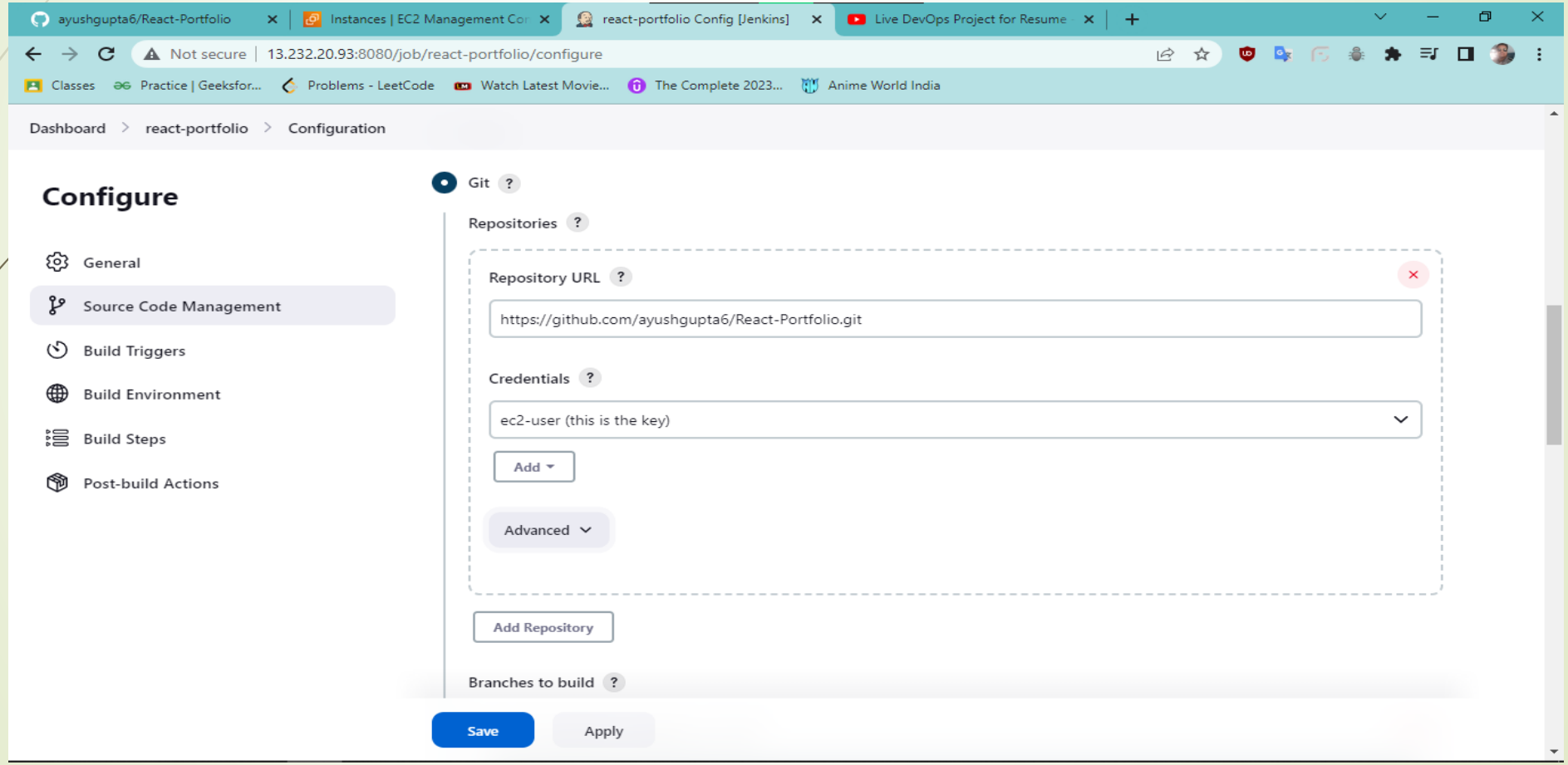
Step5.2: Key Generation and Adding

Adding Public SSH to the Github



Step5.3: Key Generation and Adding

Adding private key to the Jenkins



DockerFile Code

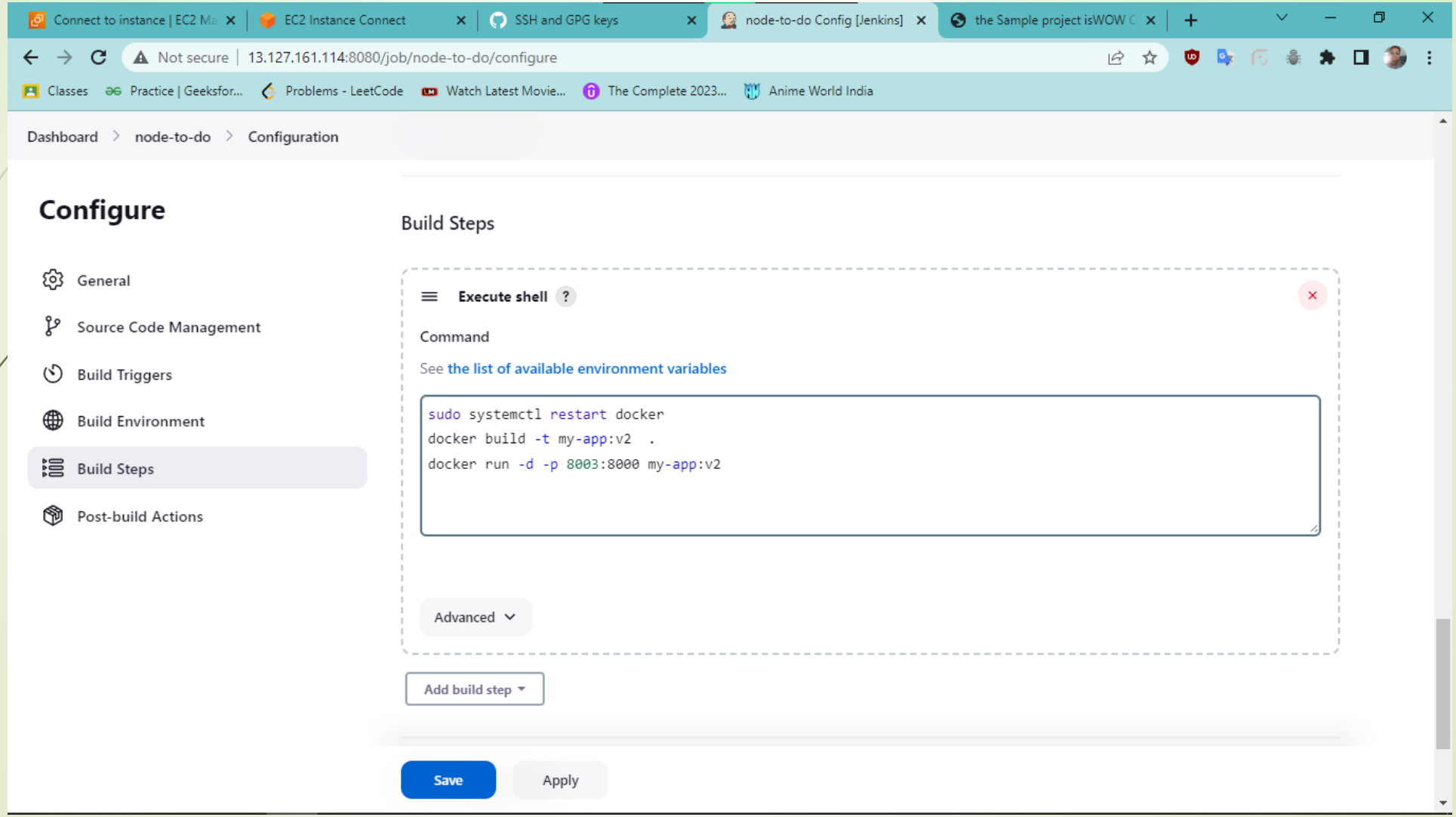
```
FROM node:12.2.0-alpine
WORKDIR app
COPY . .
RUN npm install
RUN npm run test
EXPOSE 8000
CMD ["node","app.js"]
```

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%2Fd
ocker.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_mahav
ira
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: Exectuble Shell



Step8: Console Output

node-to-do #8 Console [Jenkins]

node-to-do/8/console

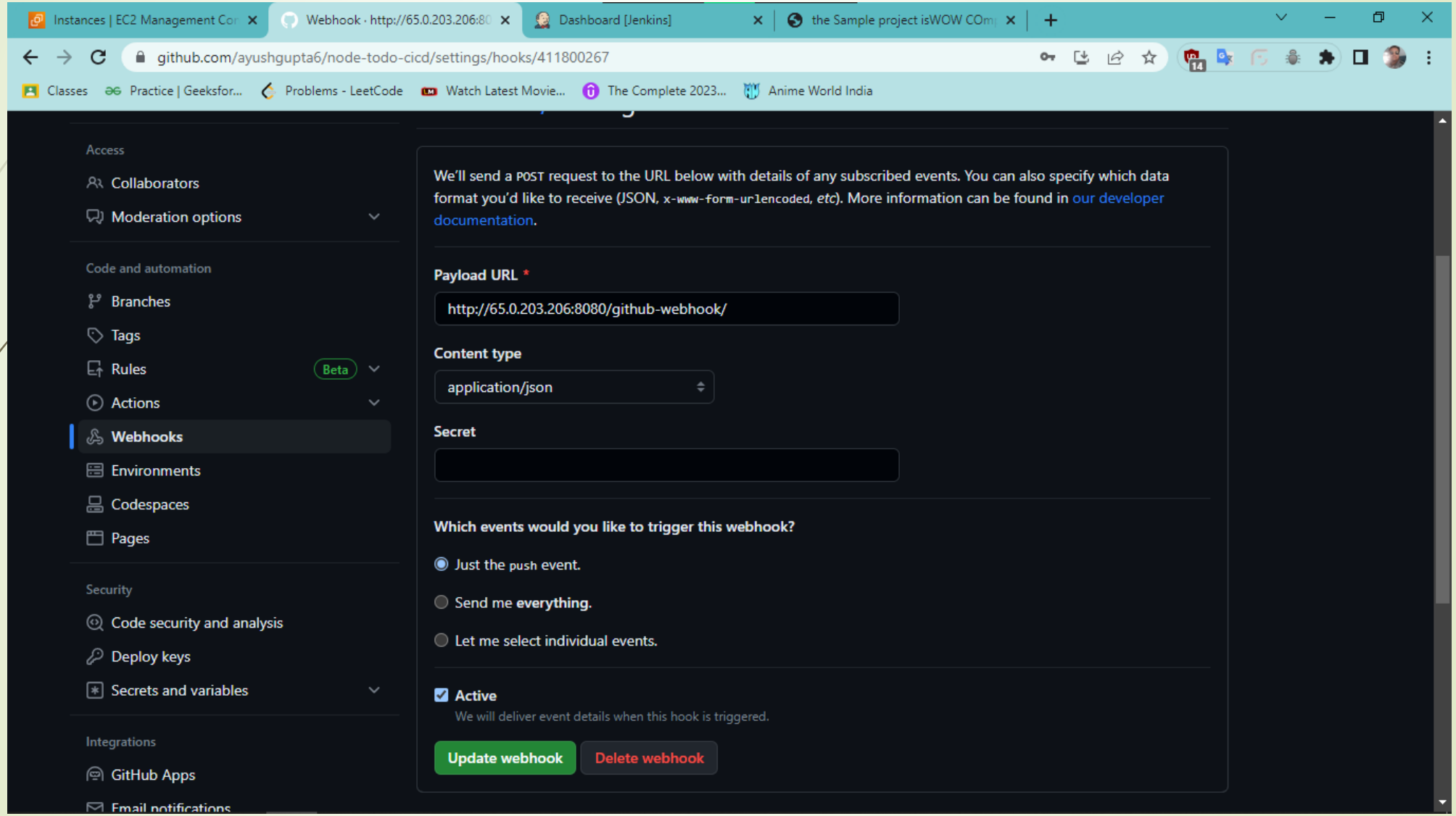
Watch Latest Movie... The Complete 2023... Anime World India

Console Output

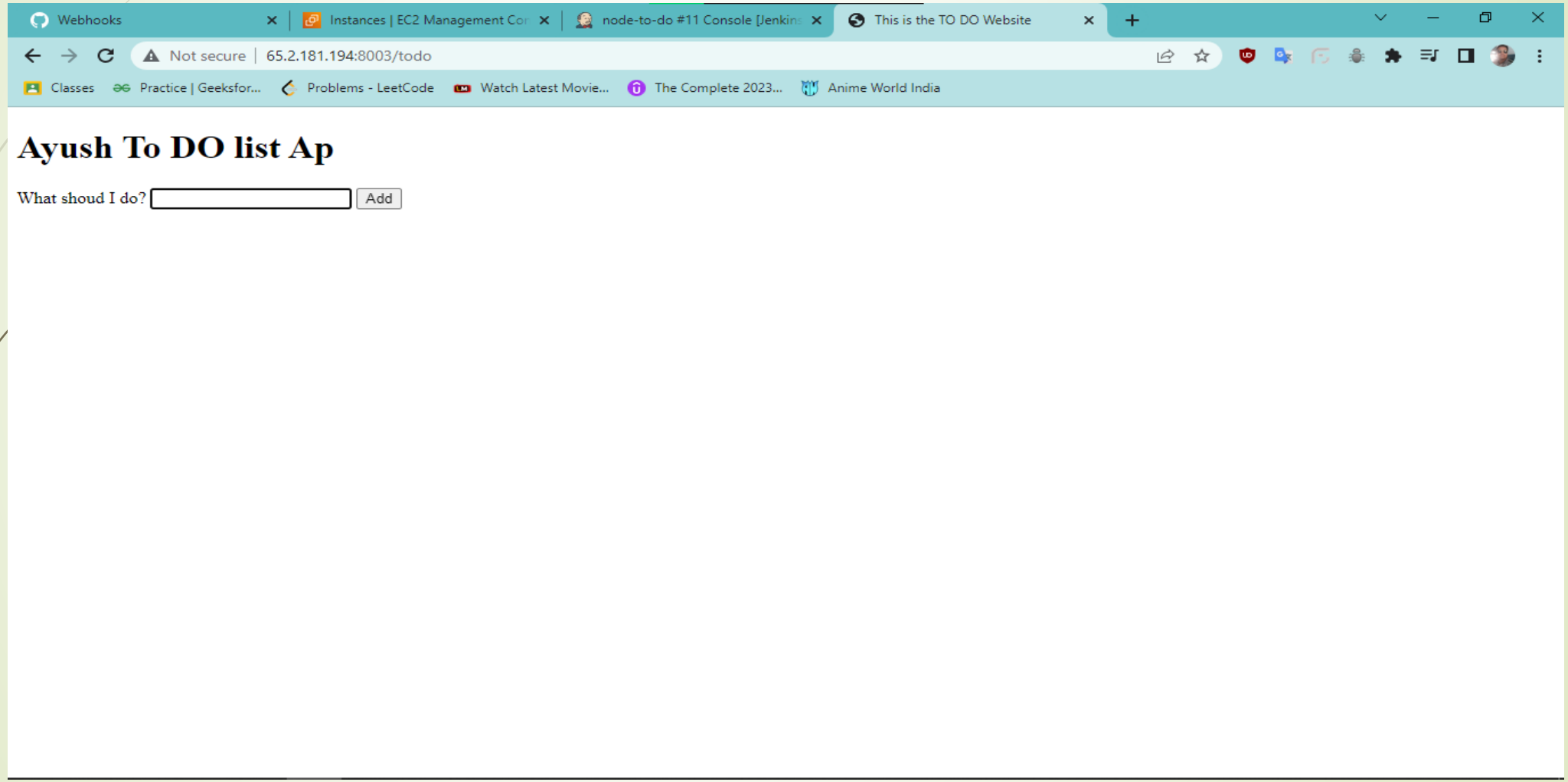
```
Started by user Ayush
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/node-to-do
The recommended git tool is: NONE
using credential jekins-sample
 > git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/node-to-do/.git # timeout=10
Fetching changes from the remote Git repository
 > 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)



Step10: WebSite



References

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

THANK YOU