D.Y.Patil International University, Akurdi, Pune

School of Computer Science Engineering and Applications (SCSEA)

Final Year Engineering (B.Tech)

Cloud Automation & Devops CSE (CLC-4001)

Lab Manual

D Y PATIL
INTERNATIONAL
UNIVERSITY
AKURDI PUNE

Vision of the University:

"To Create a vibrant learning environment – fostering innovation and creativity, experiential learning, which is inspired by research, and focuses on regionally, nationally and globally relevant areas."

Mission of the School:

- *To provide a diverse, vibrant and inspirational learning environment.*
- *To establish the university as a leading experiential learning and research oriented center.*
- *To become a responsive university serving the needs of industry and society.*
- *To embed internationalization, employability and value thinking*

Cloud Automation & Devops CSE

Course Objectives:

1. To introduce DevOps principles and automation workflows for modern cloud environments.
2. To implement CI/CD pipelines and Infrastructure as Code (IaC) practices.
3. To apply containerization and orchestration techniques using Docker and Kubernetes.
4. To integrate DevOps tools with cloud-native services across AWS/Azure/GCP
5. To monitor, manage, and optimize cloud-deployed applications effectively

Course Outcomes:

On completion of the course, learner will be able to–

CO1	**Understand DevOps culture, lifecycle, and tools relevant to cloud automation**
CO2	**Implement version control and CI/CD practices for efficient delivery**
CO3	**Utilize Infrastructure as Code (IaC) and configuration management in cloud platforms.**
CO4	**Apply containerization and orchestration techniques using Docker and Kubernetes.**
CO5	**Integrate monitoring and cloud-native DevOps tools for application lifecycle management.**

Rules and Regulations for Laboratory:

- Students should be regular and punctual to all the Lab practical
- Lab assignments and practical should be submitted within given time.
- Mobile phones are strictly prohibited in the Lab.
- Please shut down the Computers beforeleaving the Lab.
- Please switch off the fans and lights and keep the chair in proper position before leaving the Lab
- Maintain proper discipline in Lab

D.Y.Patil International University, Akurdi, Pune

School of Computer Science Engineering and Applications

Index

Sr. No.	Name of the Practical	Date of Conducti on	Page No.		Sign of Teacher	Remarks*
			From	To		
1.	Git workflow with branching, merging, and pull requests					
2.	Setting up a Jenkins CI/CD pipeline for a sample project					
3.	Containerizing applications with Docker and Docker Compose					
4.	Setup the Docker swarm installation (master slave architecture)					
5.	Web application hosting with docker stack and docker compose (Voting_app)					
6.	Deploying multi-container on Kubernetes					

***Absent/Attended/Late/Partially Completed/Completed**

D Y PATIL
INTERNATIONAL
UNIVERSITY
AKURDI PUNE

School of Computer Science, Engineering, and Applications

CERTIFICATE

This certificate acknowledges that **Mr. Chandrabhushan Lilhare** , **PRN: 20220802020**, enrolled as a student in **FIY B.Tech (CSE) Track: Cloud Computing** has satisfactorily fulfilled the Practical Lab Assignments requirements for the year **2025-26** during the **VII SEM**, contributing to the partial completion of the **Cloud Automation & Devops** subject.

Hence Certified!!

Mr. Vishwajeet Jagtap
Subject Faculty

Dr. Rahul Sharma
Director, SCSEA

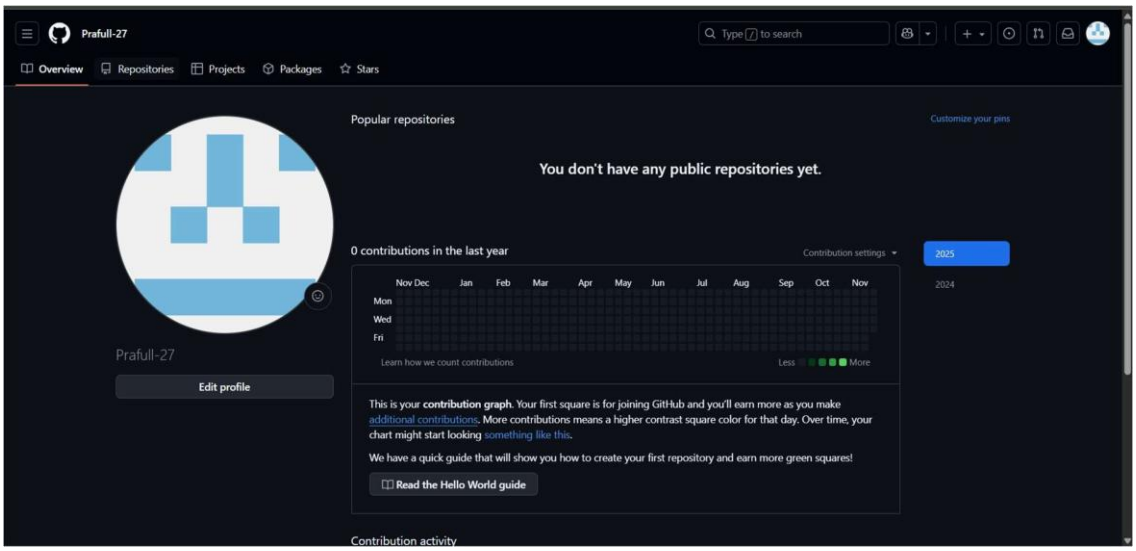
D Y PATIL INTERNATIONAL UNIVERSITY
AKURDI PUNE

School of Computer Science, Engineering and Applications(SCSEA)
B.Tech FIY (CCSA)
Subject : Cloud Automation & Devops (P)

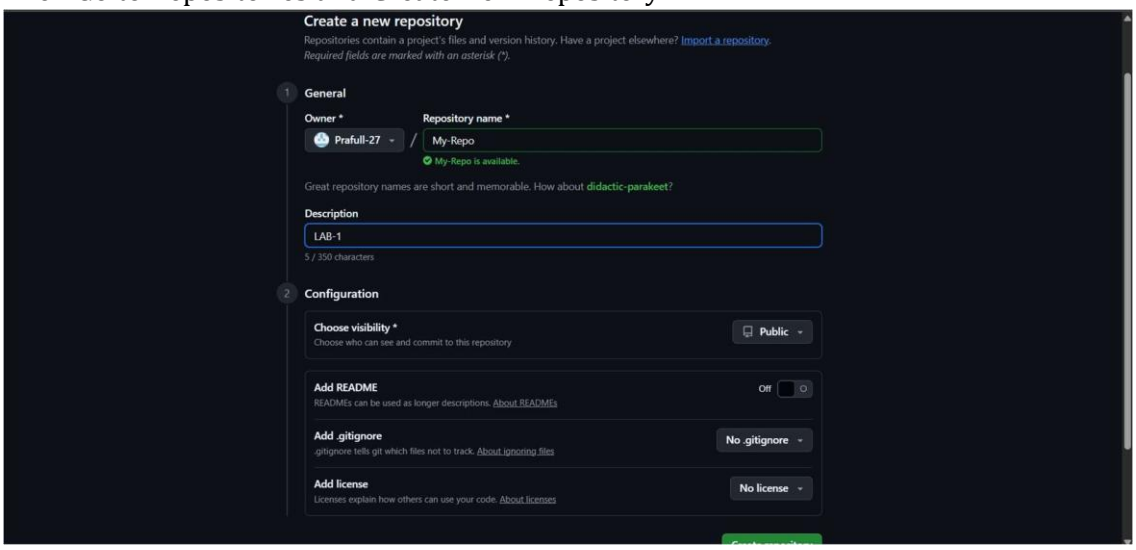
Name of the Student: **Chandrabhushan Lilhare** **PRN: 20220802020**

Title of Practicle : **1.Git workflow with branching, merging, and pull requests**

1. Create account on GitHub



2. Then Go to Repositories and Create New Repository



School of Computer Science, Engineering and Applications(SCSEA)
B.Tech FIY (CCSA)
Subject : Cloud Automation & Devops (P)

Name of the Student: **Chandrabhushan Lilhare** **PRN: 20220802020**

Title of Practicle : **1.Git workflow with branching, merging, and pull requests**

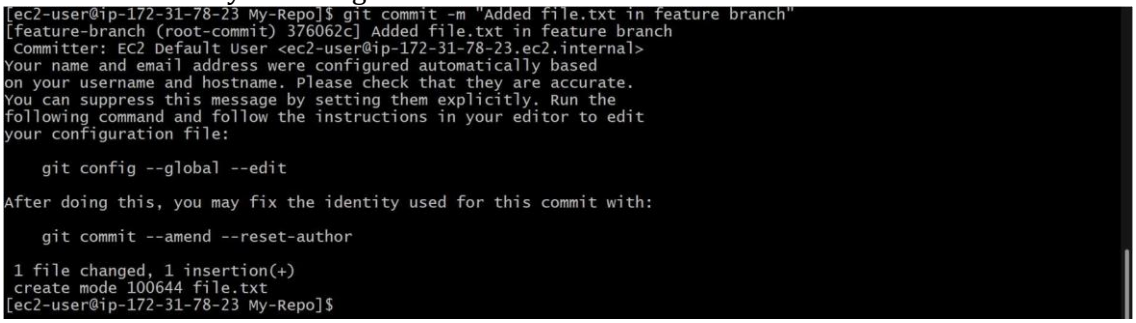
3. Now Clone This repository with Your system (ec2-instance).
   $ git clone https://github.com/Prafull-27/My-Repo.git

4. Now our pwd to repository .

5. Now Create New Branch.

6. Make changes in your branch:
   Edit or create any file.

7. Add and commit your changes



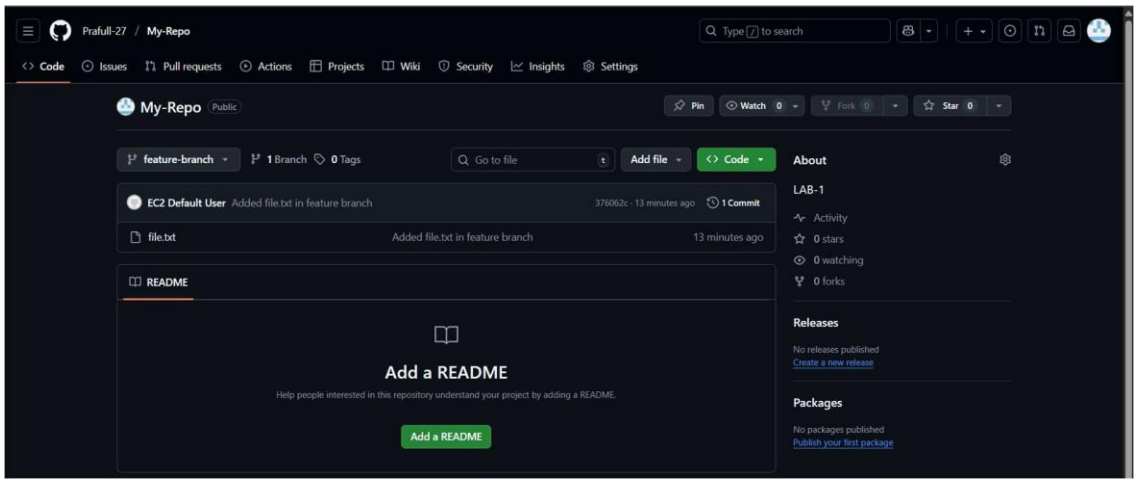
Name of the Student: **Chandrabhushan Lilhare** **PRN: 20220802020**

Title of Practicle : **1.Git workflow with branching, merging, and pull requests**

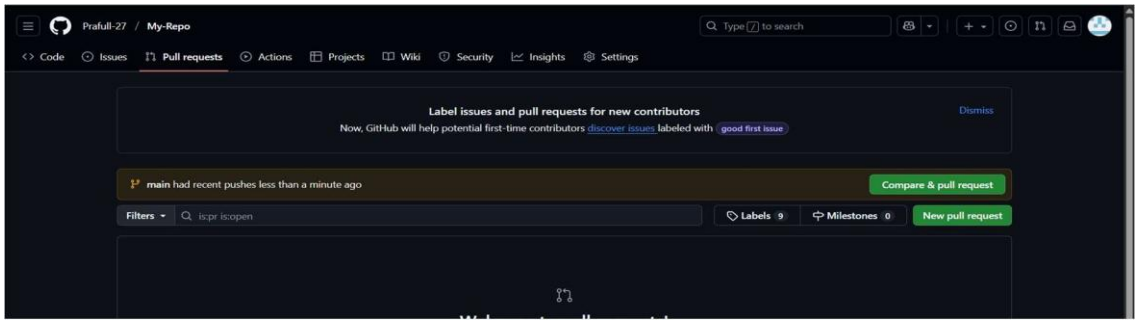
8. Push your branch to GitHub

```
[ec2-user@ip-172-31-78-23 My-Repo]$ git push -u origin feature-branch
Username for 'https://github.com': Prafull-27
Password for 'https://Prafull-27@github.com':
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 263 bytes | 263.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Prafull-27/My-Repo.git
 * [new branch]      feature-branch -> feature-branch
branch 'feature-branch' set up to track 'origin/feature-branch'.
[ec2-user@ip-172-31-78-23 My-Repo]$ ls
file.txt
```

9. Create Pull Request (PR)
   Go to your GitHub repo



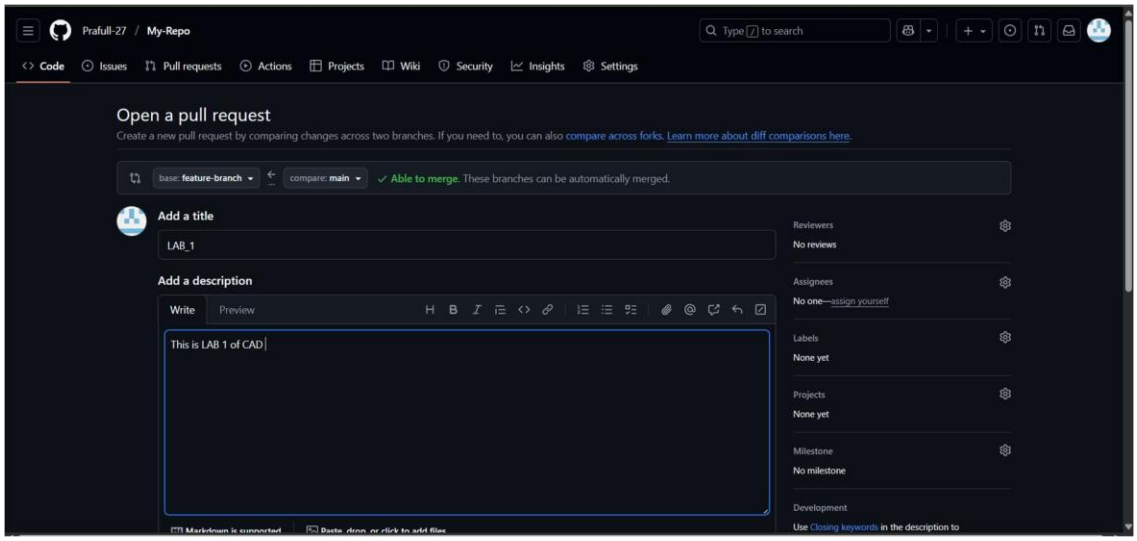
You will see a **Compare & Pull Request** button → click it



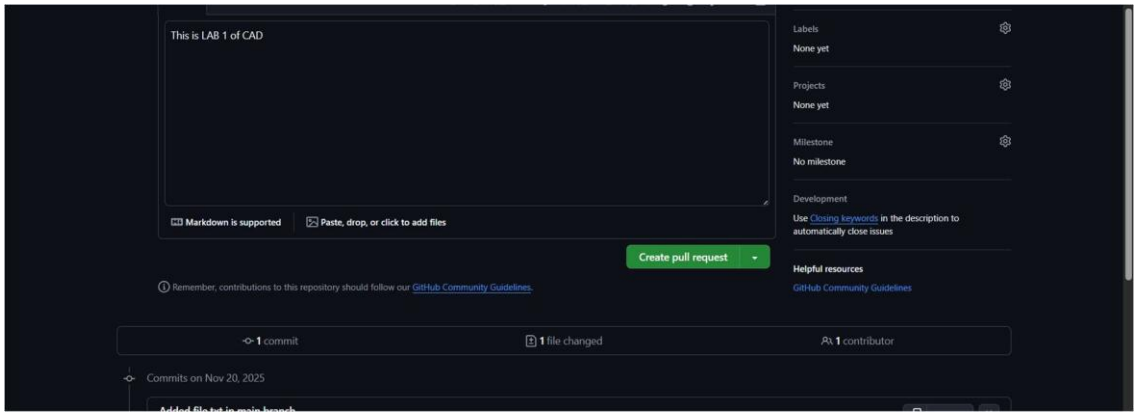
Name of the Student: **Chandrabhushan Lilhare** **PRN: 20220802020**

Title of Practicle : **1.Git workflow with branching, merging, and pull requests**

Add a title and description



Click Create Pull Request



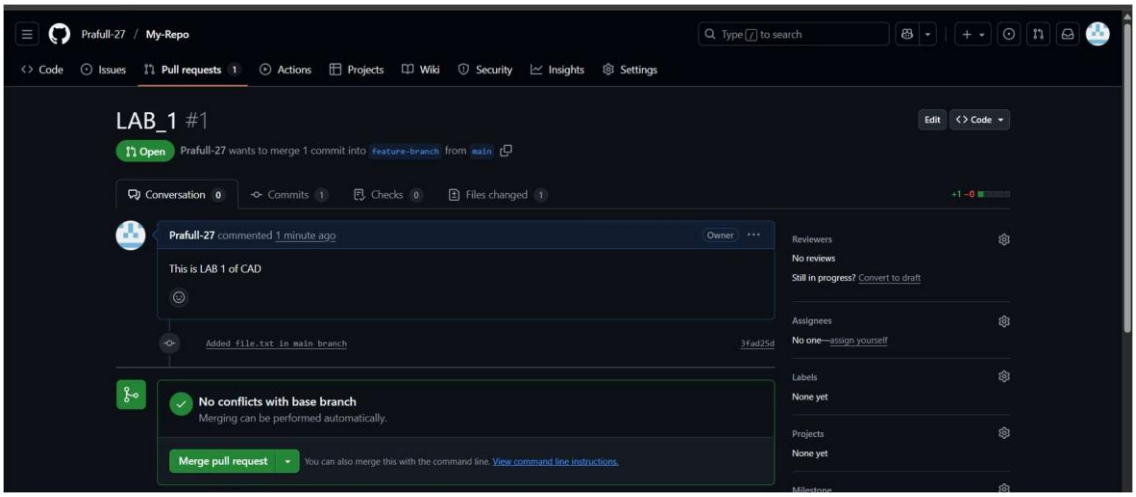
Name of the Student: **Chandrabhushan Lilhare** **PRN: 20220802020**

Title of Practicle : **1.Git workflow with branching, merging, and pull requests**

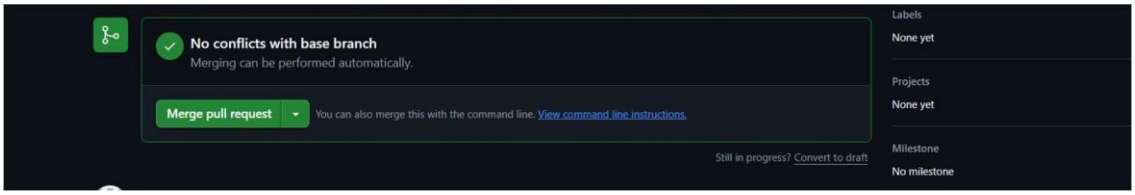
10. Review and Merge PR

On GitHub:

1. Open the PR



2. Click **Merge Pull Request**



3. Then click **Confirm Merge**

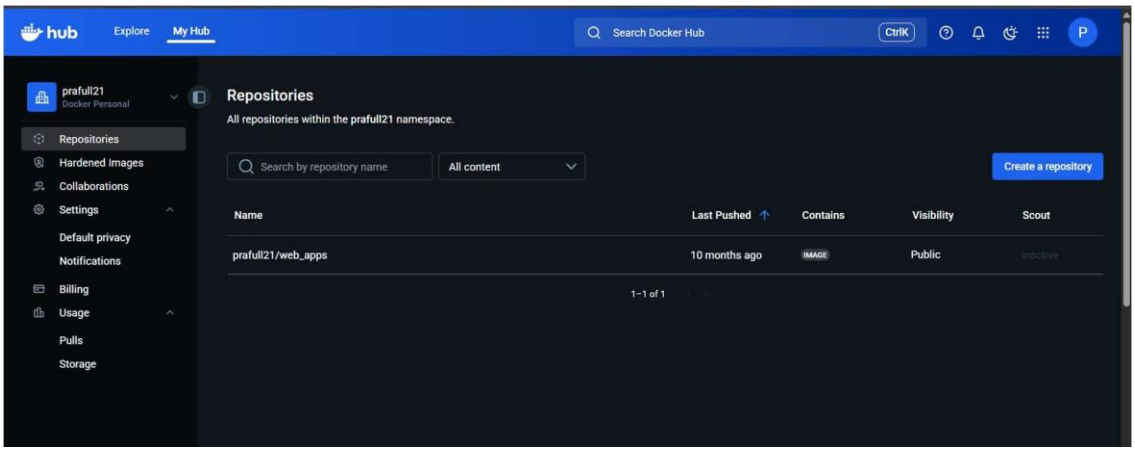
D Y PATIL INTERNATIONAL UNIVERSITY
AKURDI PUNE

School of Computer Science, Engineering and Applications(SCSEA)
B.Tech FIY (CCSA)
Subject : Cloud Automation & Devops (P)

Name of the Student: **Chandrabhushan Lilhare** **PRN: 20220802020**

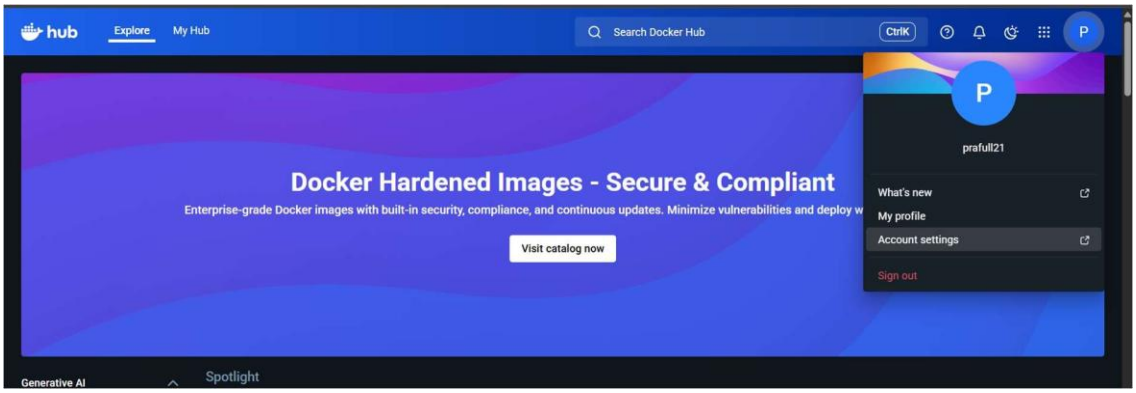
Title of Practicle : **2. Setting up a Jenkins CI/CD pipeline for a sample project**

Step 1 Create Docker Hub account.



Step 2 — Generate Docker Hub Access Token

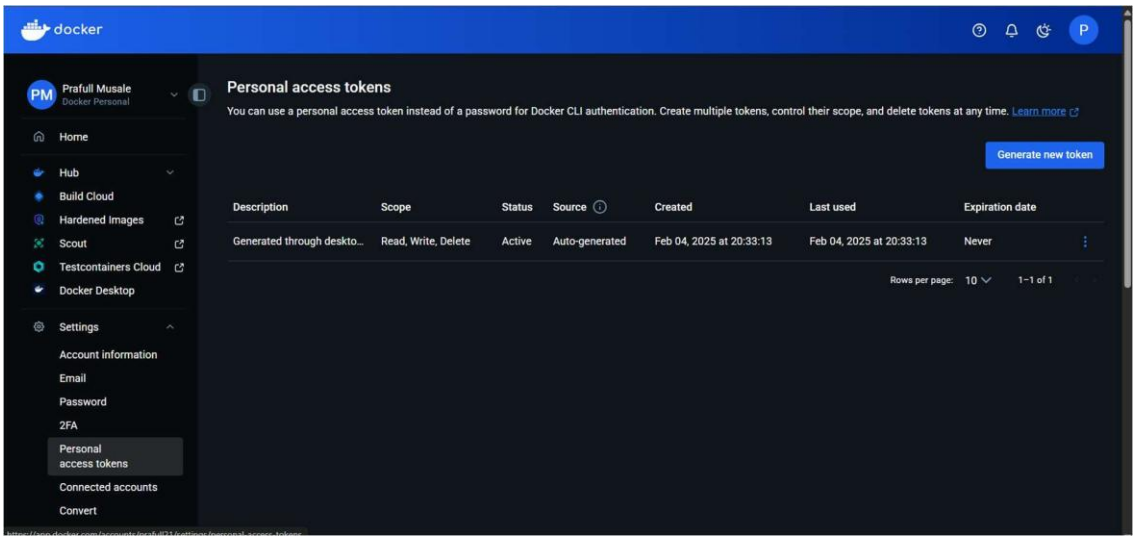
1. Go to **Account Settings**



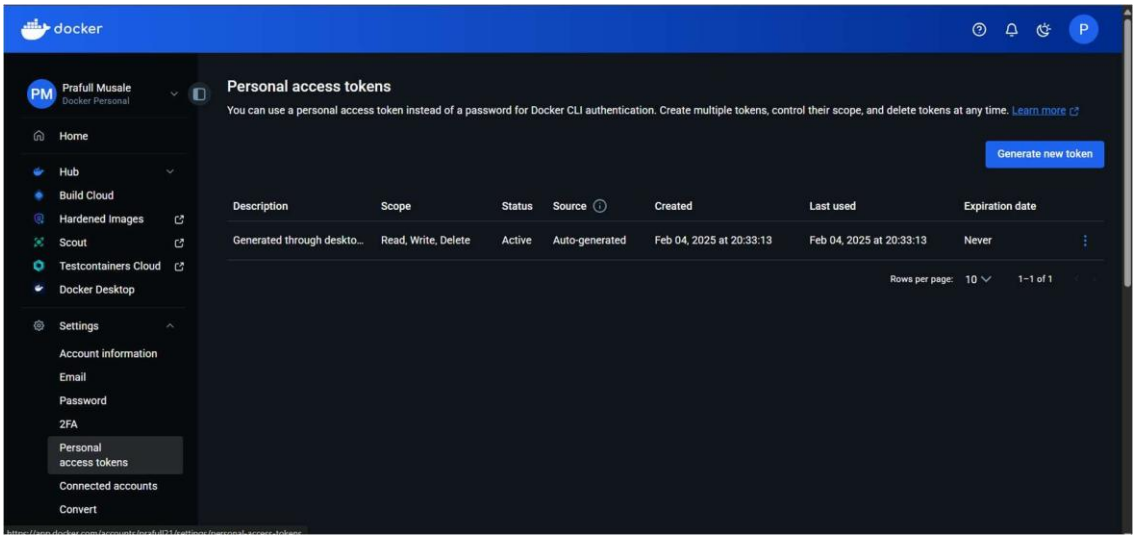
Name of the Student: Chandrabhushan Lilhare **PRN: 20220802020**

Title of Practicle : **2. Setting up a Jenkins CI/CD pipeline for a sample project**

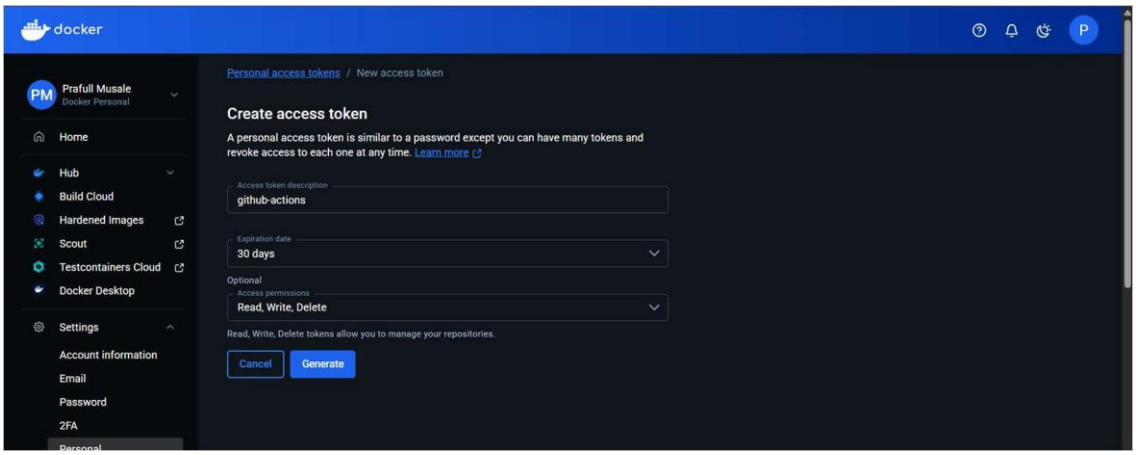
2. Click **Person Access Token**



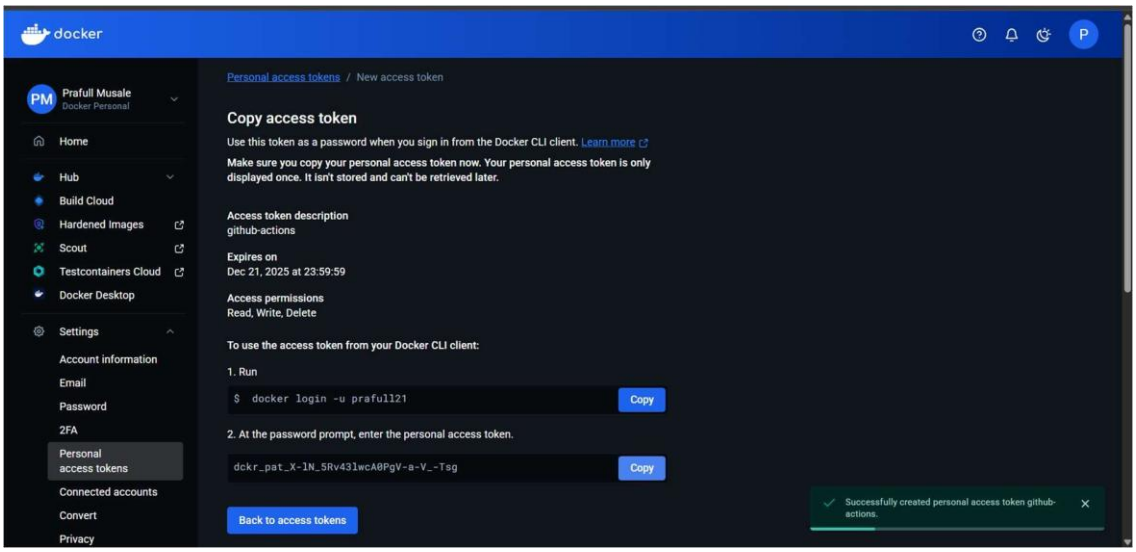
3. Click **New Access Token**



4. Name: github-actions

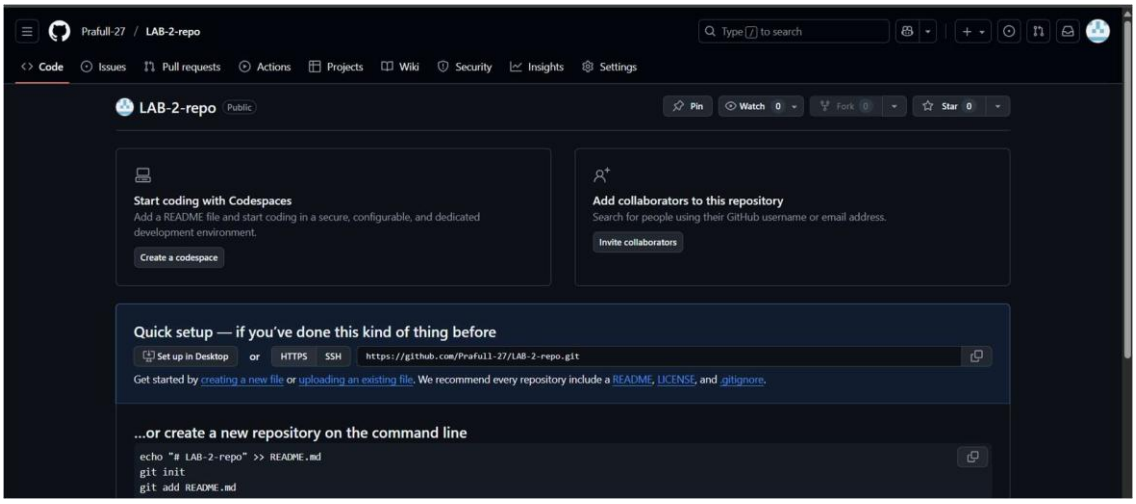


5. Copy the token (you will use it soon)



Step 3 — Add Secrets in GitHub Repository

1. Open your repository



2. Go to:
   Settings → Secrets and variables → Actions → New repository secret

Create two secrets:

Secret 1:

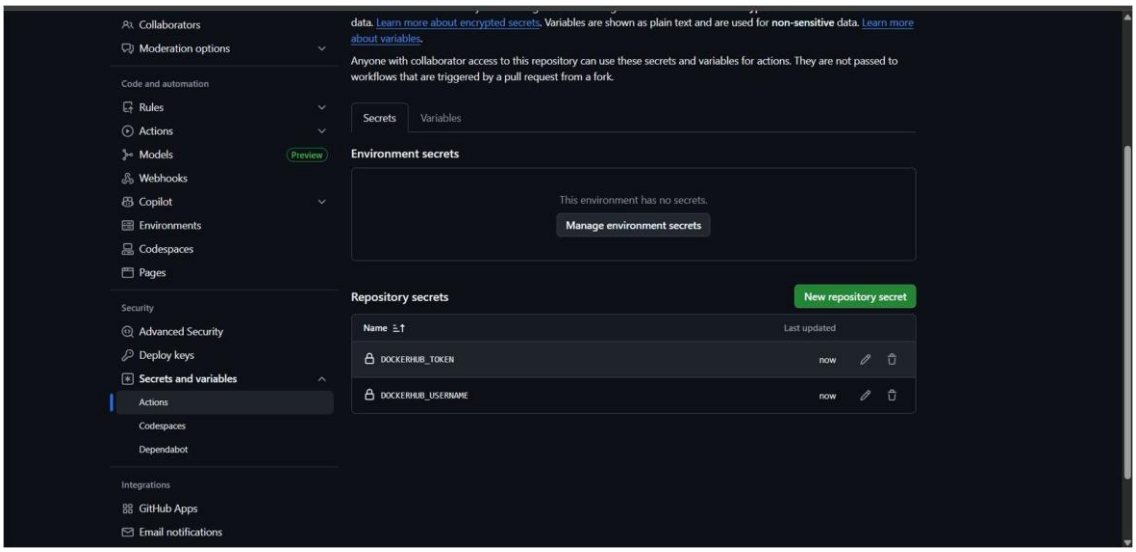
- Name: DOCKERHUB_USERNAME
- Value: your Docker Hub username

Secret 2:

- Name: DOCKERHUB_TOKEN
- Value: the token you generated in Step 2

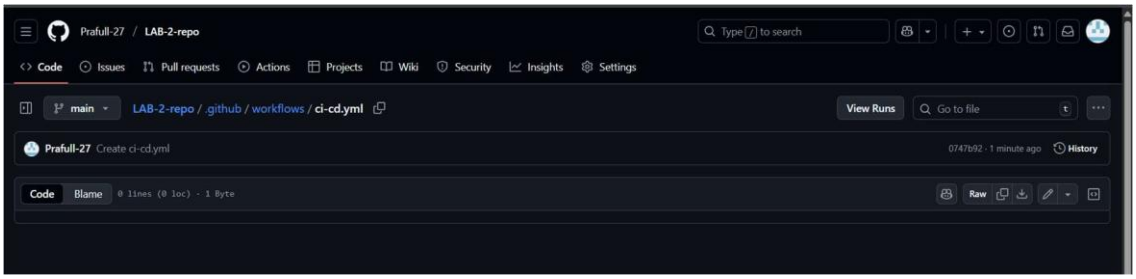
Name of the Student: Chandrabhushan Lilhare **PRN: 20220802020**

Title of Practicle : **2. Setting up a Jenkins CI/CD pipeline for a sample project**



Step 4 — Create Workflow Folder

In your repository, create this folder: .github/workflows/



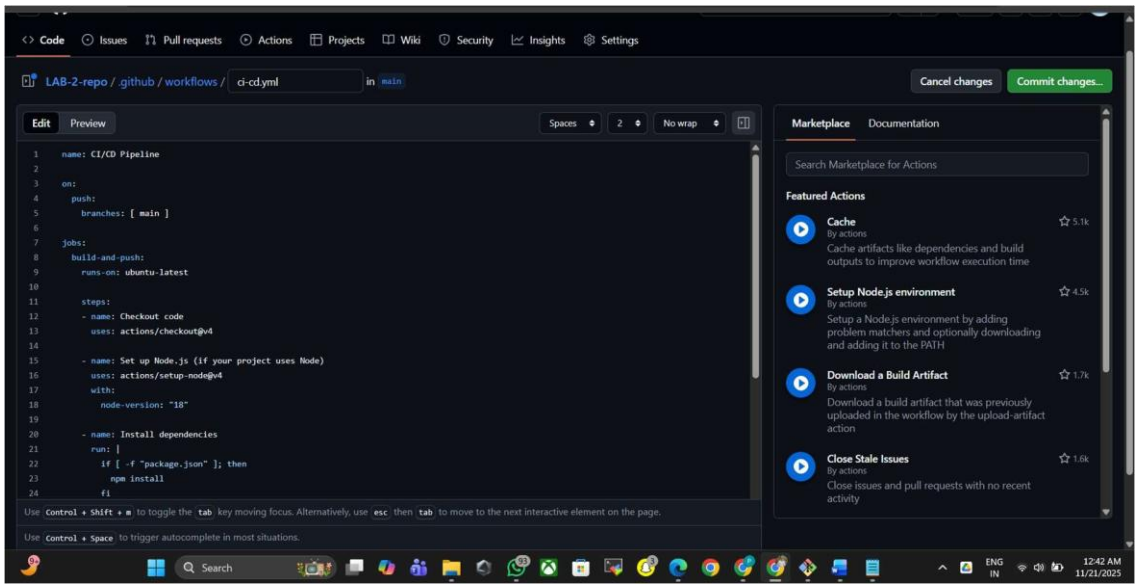
Step 5 — Create Workflow File (CI/CD File)

Inside .github/workflows/, create a file

Paste this code inside (simple CI/CD pipeline)

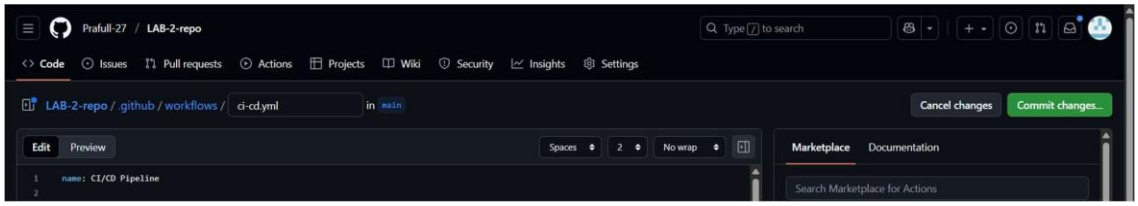
Name of the Student: Chandrabhushan Lilhare **PRN: 20220802020**

Title of Practicle : **2. Setting up a Jenkins CI/CD pipeline for a sample project**



Step 6 — Commit & Push

If you push to the **main branch**, the pipeline runs automatically.

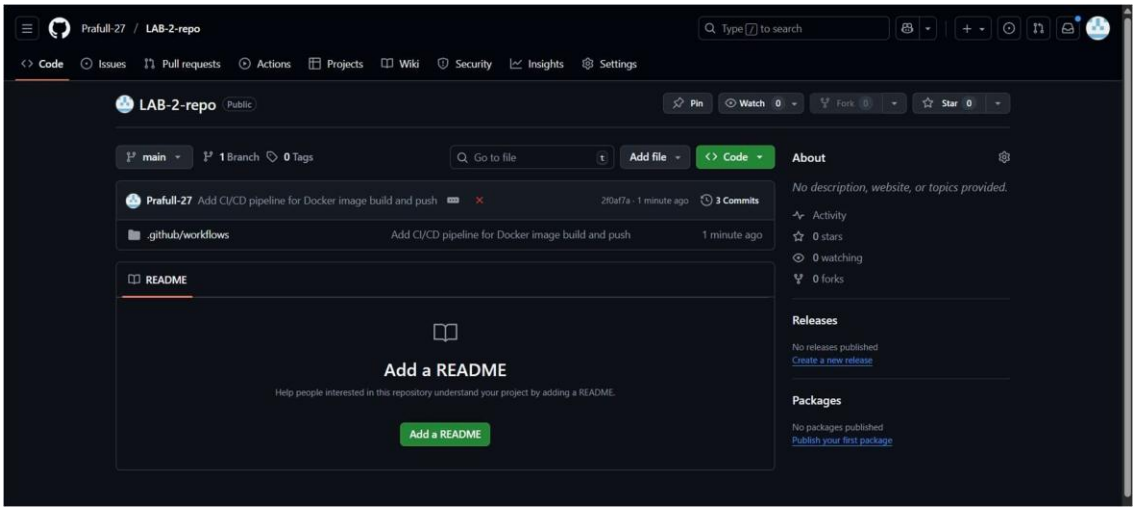


Name of the Student: Chandrabhushan Lilhare **PRN: 20220802020**

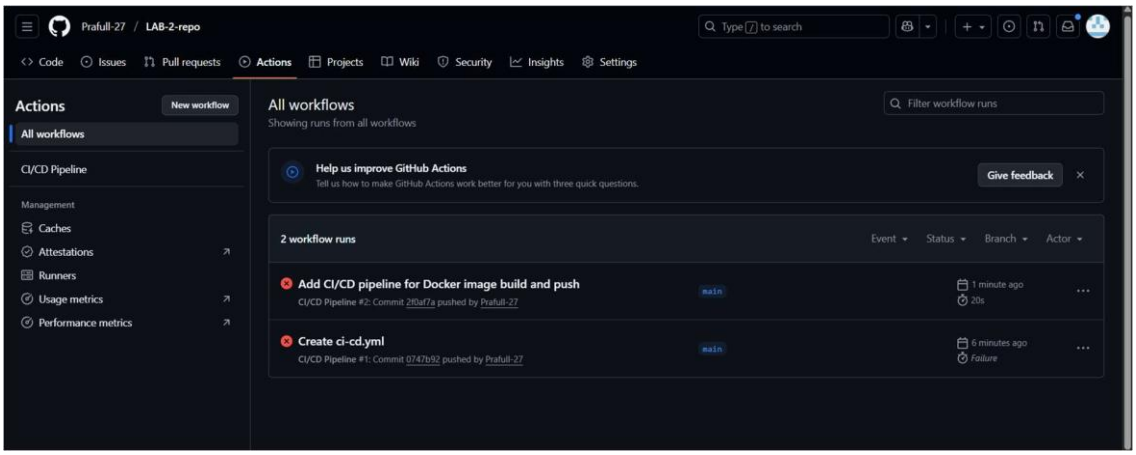
Title of Practicle : **2. Setting up a Jenkins CI/CD pipeline for a sample project**

Step 7 — Check the GitHub Actions Pipeline

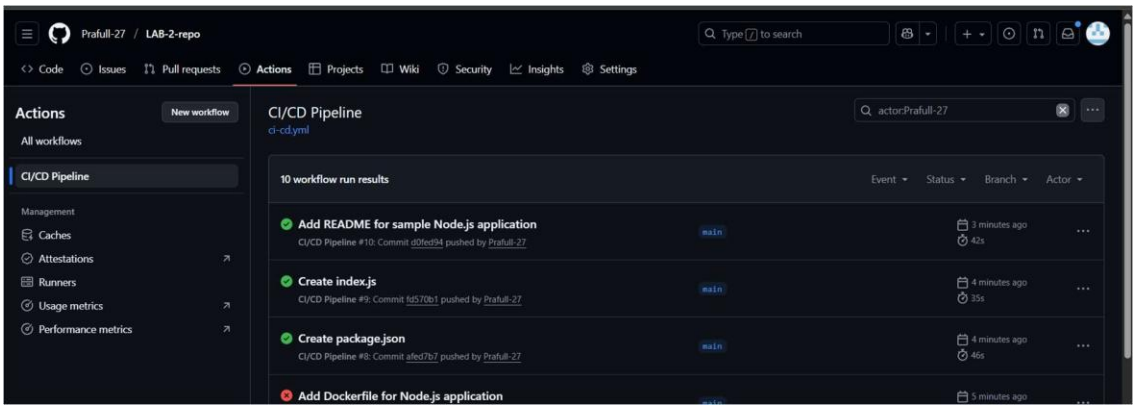
1. Go to your repository



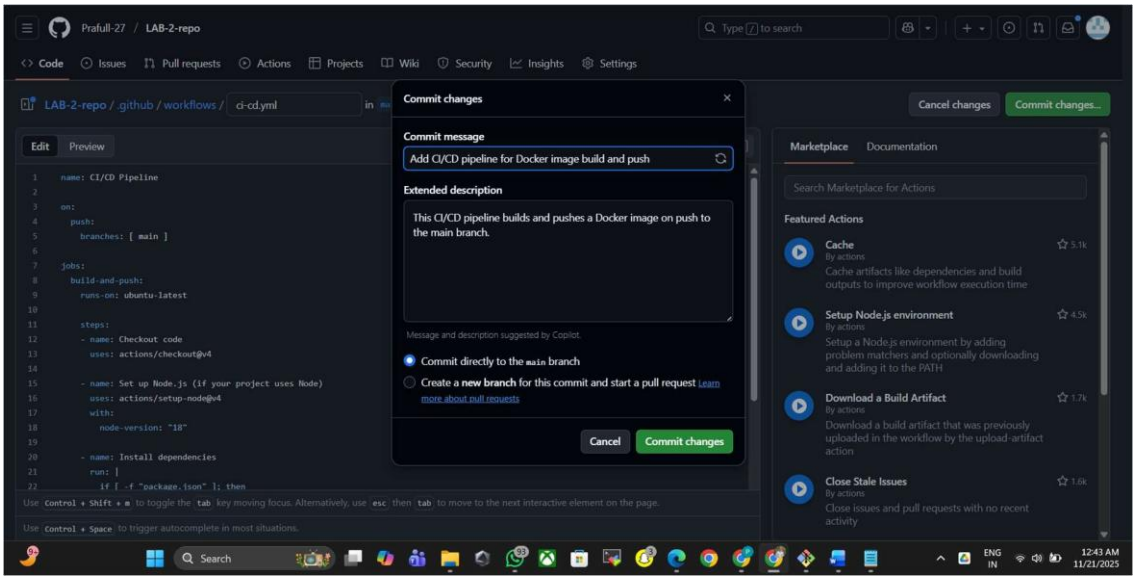
2. Click **Actions**



3. You will see a new workflow running



4. Click it → you can see all logs: build, test, docker push, etc.



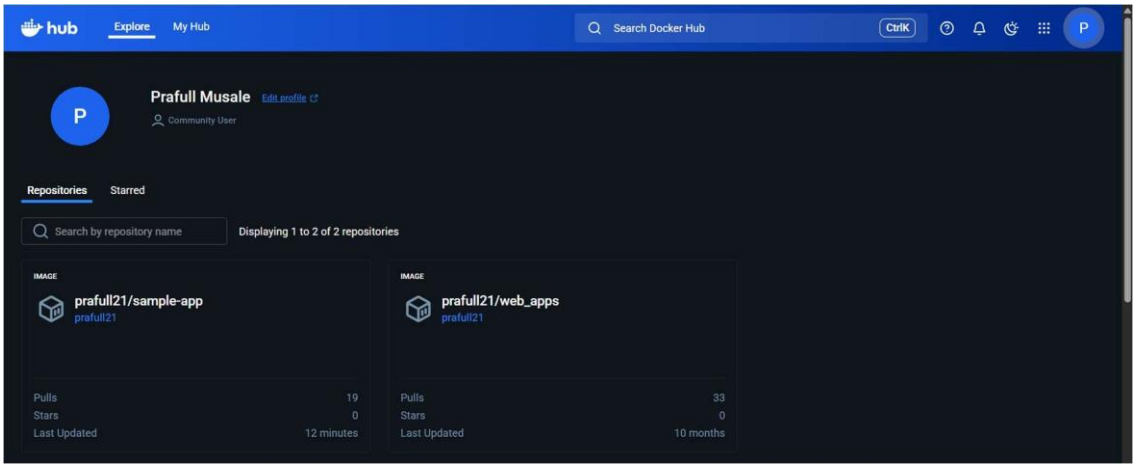
Name of the Student: Chandrabhushan Lilhare **PRN: 20220802020**

Title of Practicle : **2. Setting up a Jenkins CI/CD pipeline for a sample project**

Step 8 — Confirm Docker Image in Docker Hub

Go to Docker Hub → Repositories →
You will see:

- sample-app:latest
- sample-app:<build-number>



D Y PATIL INTERNATIONAL UNIVERSITY
AKURDI PUNE

School of Computer Science, Engineering and Applications(SCSEA)
B.Tech FIY (CCSA)
Subject : Cloud Automation & Devops (P)

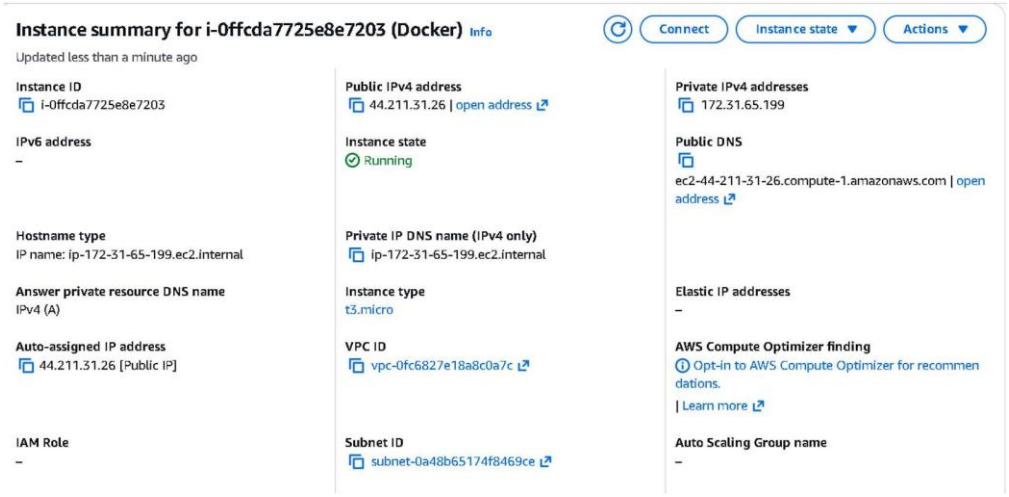
Name of the Student:	**Chandrabhushan Lilhare**	**PRN: 20220802020**
Title of Practicle :	**3. Containerizing applications with Docker and Docker Compose**	

1. Launch the instance.

Name: Docker

AMI: Amazon Linux2023

Instance Type: t3.micro



2. Install the Docker & Docker Compose.

sudo yum update -y

sudo dnf install docker -y

sudo systemctl start docker

sudo systemctl enable docker

sudo usermod -aG docker ec2-user

sudo usermod -aG docker $USER

sudo curl -L "https://github.com/docker/compose/releases/download/v2.29.2/docker-compose-linux-x86_64" -o /usr/local/bin/docker-compose

Name of the Student: **Chandrabhushan Lilhare** **PRN: 20220802020**

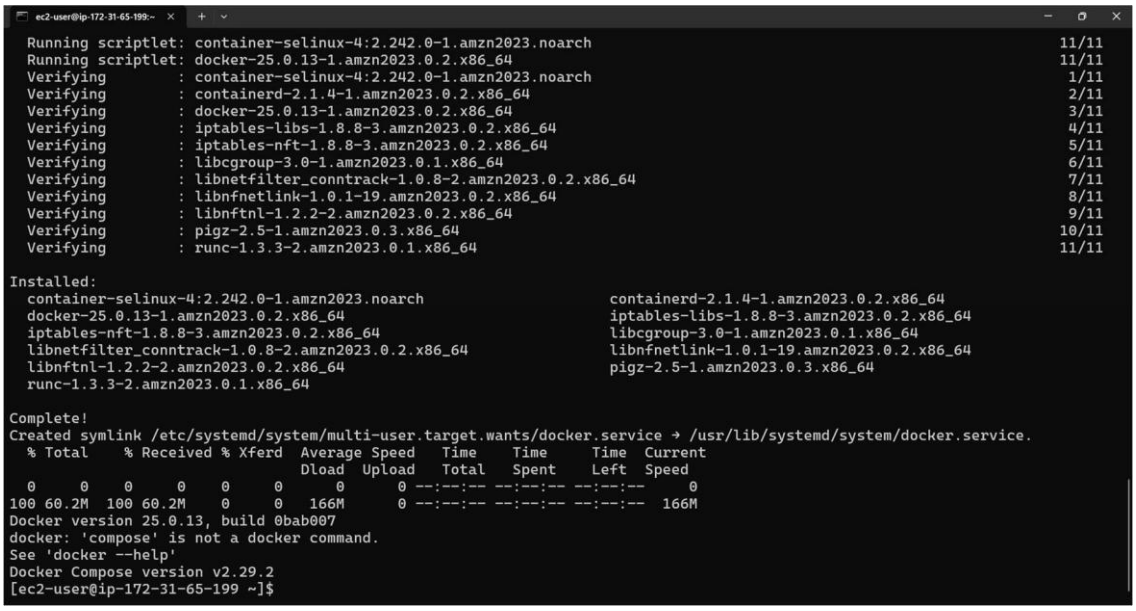
Title of Practicle : **3. Containerizing applications with Docker and Docker Compose**

sudo chmod +x /usr/local/bin/docker-compose

docker --version

docker compose version

docker-compose version



Name of the Student: Chandrabhushan Lilhare **PRN: 20220802020**

Title of Practicle : 3. Containerizing applications with Docker and Docker Compose

3. Create Your Project Folder

mkdir myapp

cd myapp

4. Add Your Application Code

Example: Node.js Express app

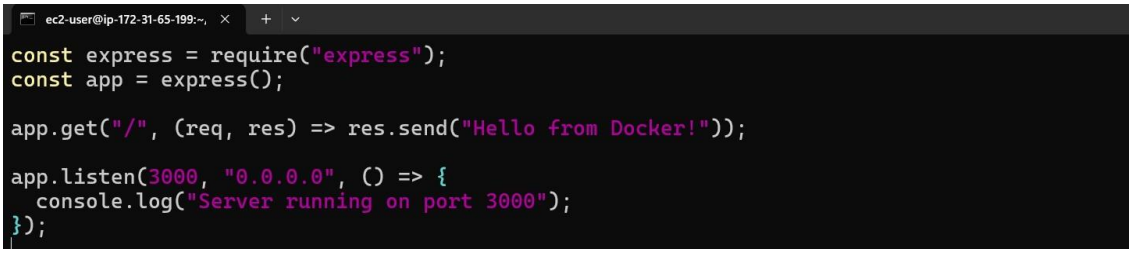
Create app.js

$ vi app.js

```
const express = require("express");

const app = express();


app.get("/", (req, res) => res.send("Hello from Docker!"));


app.listen(3000, () => console.log("Server running on port 3000"));
```

Name of the Student: Chandrabhushan Lilhare **PRN: 20220802020**

Title of Practicle : **3. Containerizing applications with Docker and Docker Compose**



Create package.json

$ package.json

{

"name": "docker-app",

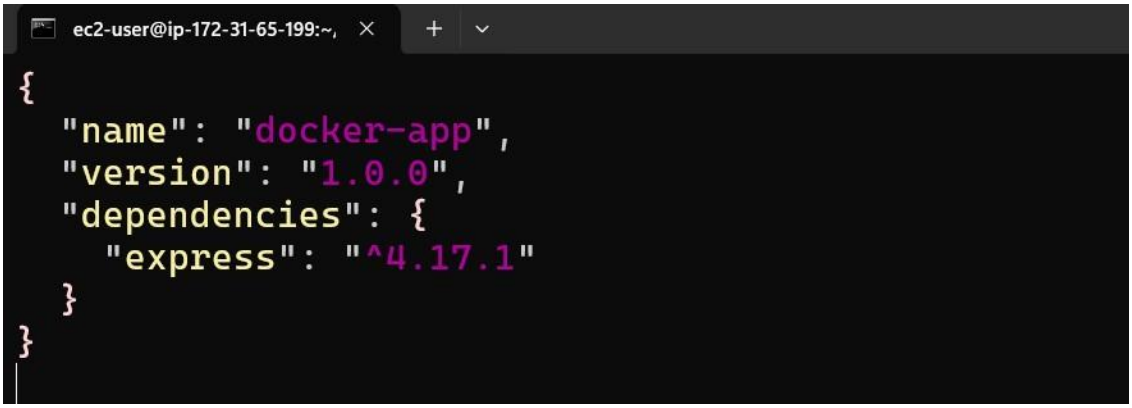
"version": "1.0.0",

"dependencies": {

"express": "^4.17.1"

}

}



Name of the Student: Chandrabhushan Lilhare **PRN: 20220802020**

Title of Practicle : 3. Containerizing applications with Docker and Docker Compose

5. Write the Dockerfile

Create Dockerfile

$ vi dockerfile

FROM node:18

WORKDIR /app

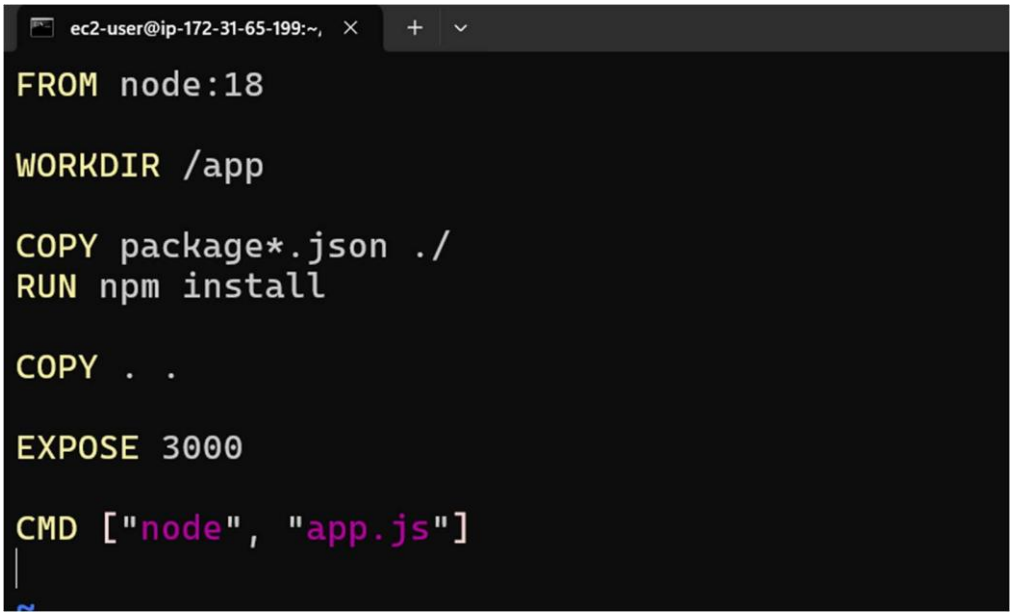
COPY package*.json ./

RUN npm install

COPY . .

EXPOSE 3000

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

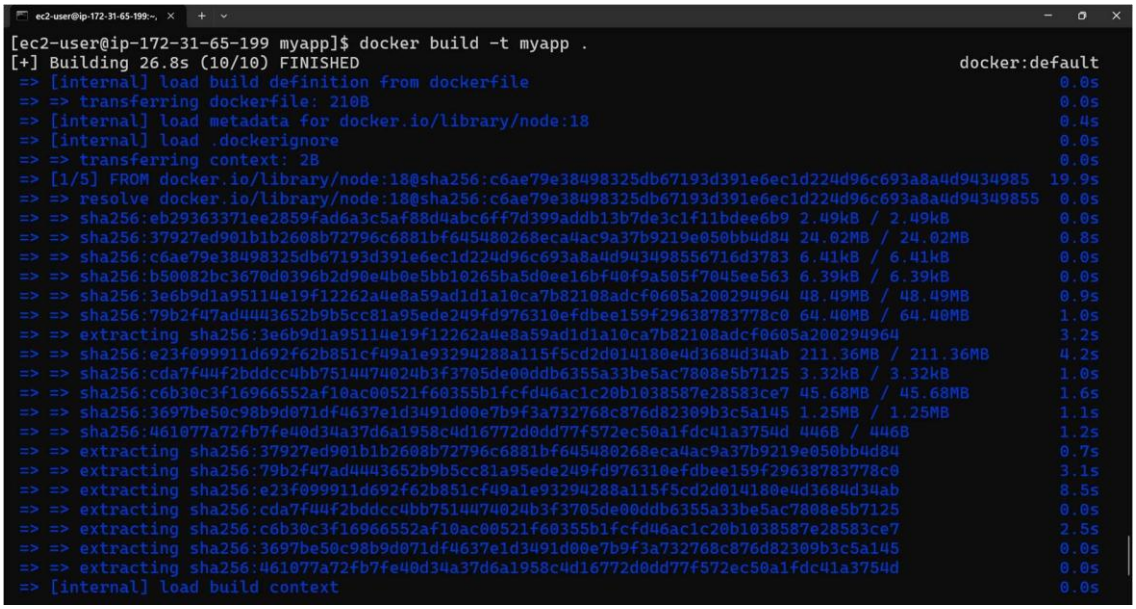


Name of the Student: Chandrabhushan Lilhare **PRN: 20220802020**

Title of Practicle : 3. Containerizing applications with Docker and Docker Compose

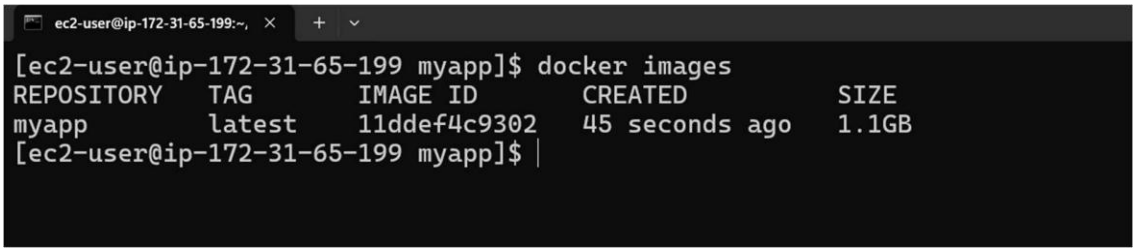
6. Build Docker Image

$ docker build -t myapp .



Check image:

$ docker images

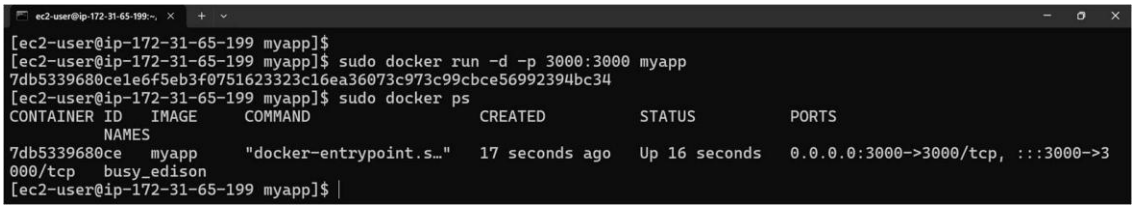


Name of the Student: Chandrabhushan Lilhare **PRN: 20220802020**

Title of Practicle : 3. Containerizing applications with Docker and Docker Compose

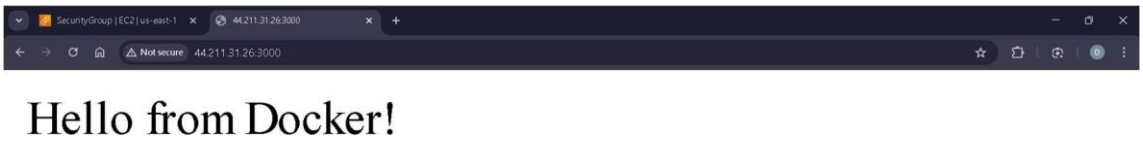
7. Run Container (Without Compose)

$ docker run -d -p 3000:3000 --name myapp_container myapp



Test:

http://<ip>:3000



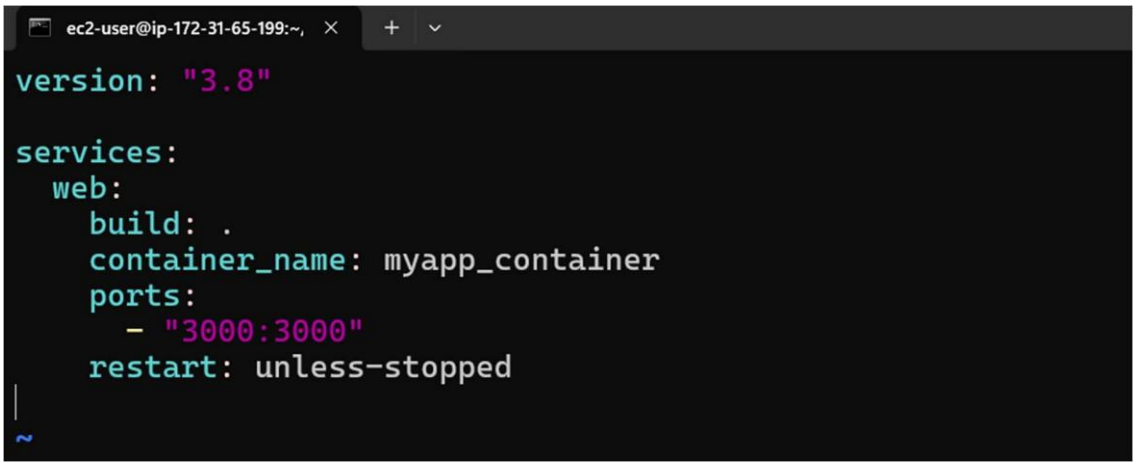
8. Create docker-compose.yml

$ docker-compose.yml

```
version: "3.8"
services:
  web:
    build: .
    container_name: myapp_container
    ports:
      - "3000:3000"
    restart: unless-stopped
```

Name of the Student: Chandrabhushan Lilhare **PRN: 20220802020**

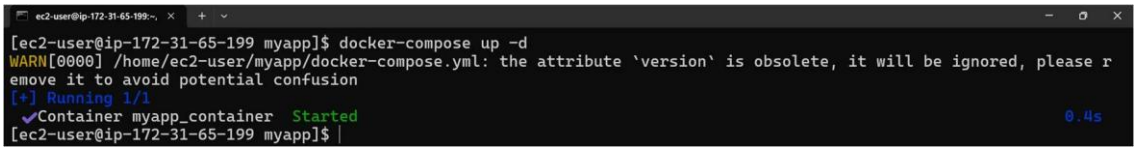
Title of Practicle : 3. Containerizing applications with Docker and Docker Compose



9. Run Using Docker Compose

Start:

$ docker-compose up -d



Check running:

$ docker ps

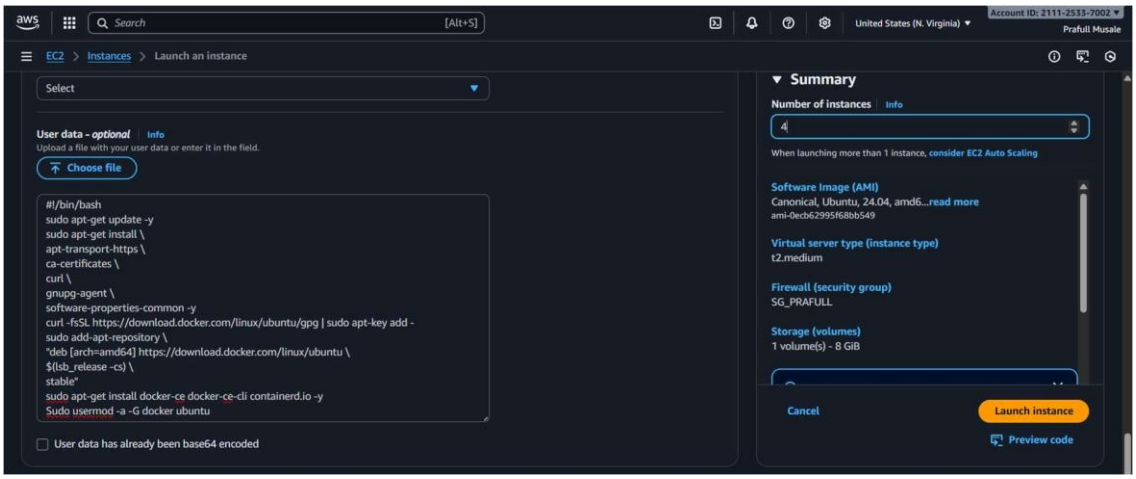
D Y PATIL INTERNATIONAL UNIVERSITY
AKURDI PUNE

School of Computer Science, Engineering and Applications(SCSEA)
B.Tech FIY (CCSA)
Subject : Cloud Automation & Devops (P)

Name of the Student: **Chandrabhushan Lilhare** **PRN: 20220802020**

Title of Practicle : **4.** Setup the Docker swarm installation (master slave architecture)

1. Launch 4 instance with docker installation



2. Connect all instance to terminal

3. Now, Fire $ sudo docker swarm init at 1st instance for give him leadership

```
ubuntu@ip-172-31-24-195:~$ sudo docker swarm init
Swarm initialized: current node (xr2iq9wci3jygl8orvt34m3nx) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-3qcxdb5nvejucsv2csp940enlhhc333wlxzyj35vu54qr0bf0k-7oubd4f7165zxrw417m28b3qj 172.31.24.195:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.
ubuntu@ip-172-31-24-195:~$
```

Name of the Student: **Chandrabhushan Lilhare** **PRN: 20220802020**

Title of Practicle : **4.** Setup the Docker swarm installation (master slave architecture)

4. Now generate token for manager and worker with following commands

$ docker swarm join-token manager

$ docker swarm join-token worker

```
ubuntu@ip-172-31-24-195:~$ docker swarm join-token manager
To add a manager to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-3qcxdb5nvejucsv2csp940enlhhc333wlxzyj35vu54qr0bf0k-5y9s
zo95v0z27sbircs2dxp4v 172.31.24.195:2377

ubuntu@ip-172-31-24-195:~$ docker swarm join-token worker
To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-3qcxdb5nvejucsv2csp940enlhhc333wlxzyj35vu54qr0bf0k-7oub
d4f7165zxrw417m28b3qj 172.31.24.195:2377

ubuntu@ip-172-31-24-195:~$
```

5. Now join 2nd instance as a manger with the help of token

Copy manager token and paste into 2nd instance

```
ubuntu@ip-172-31-18-27:~$ docker swarm join --token SWMTKN-1-3qcxdb5nvejucsv2csp940enlhhc333wlxzyj35vu54q
r0bf0k-5y9szo95v0z27sbircs2dxp4v 172.31.24.195:2377
This node joined a swarm as a manager.
ubuntu@ip-172-31-18-27:~$
```

6. Join 3rd and 4th instance as a worker with the help of token

Copy worker token and paste into 3rd and 4th instance

```
ubuntu@ip-172-31-25-4:~$ docker swarm join --token SWMTKN-1-3qcxdb5nvejucsv2csp940enlh
hc333wlxzyj35vu54qr0bf0k-7oubd4f7165zxrw417m28b3qj 172.31.24.195:2377
This node joined a swarm as a worker.
ubuntu@ip-172-31-25-4:~$
```

Name of the Student: **Chandrabhushan Lilhare** **PRN: 20220802020**

Title of Practicle : **4.** Setup the Docker swarm installation (master slave architecture)

```
ubuntu@ip-172-31-20-92:~$ docker swarm join --token SWMTKN-1-3qcxdb5nvejucsv2csp940enl
hhc333wlxzyj35vu54qr0bf0k-7oubd4f7165zxrw417m28b3qj 172.31.24.195:2377
This node joined a swarm as a worker.
ubuntu@ip-172-31-20-92:~$
```

7. Now check which node is worker and which node is manager with the help of following command . This command fire on manager node because only manager or leader can show this .

$ docker node ls

```
ubuntu@ip-172-31-24-195:~$ docker node ls
ID                            HOSTNAME            STATUS    AVAILABILITY   MANAGER STATUS   ENG
INE VERSION
z0fyvuh1hyt06dz676oly2e8m     ip-172-31-18-27     Ready     Active         Reachable        29.
0.2
aodb8f1fbdguax83e4xyxrrmq     ip-172-31-20-92     Ready     Active                          29.
0.2
xr2iq9wci3jygl8orvt34m3nx *   ip-172-31-24-195    Ready     Active         Leader           29.
0.2
atiwsk98cxw05ydhitdxuac1s     ip-172-31-25-4      Ready     Active                          29.
0.2
ubuntu@ip-172-31-24-195:~$
```

8. Now same command fire on worker node

Check worker has access or not

```
ubuntu@ip-172-31-25-4:~$ docker swarm join --token SWMTKN-1-3qcxdb5nvejucsv2csp940enlh
hc333wlxzyj35vu54qr0bf0k-7oubd4f7165zxrw417m28b3qj 172.31.24.195:2377
This node joined a swarm as a worker.
ubuntu@ip-172-31-25-4:~$ sudo docker node ls
Error response from daemon: This node is not a swarm manager. Worker nodes can't be us
ed to view or modify cluster state. Please run this command on a manager node or promo
te the current node to a manager.
ubuntu@ip-172-31-25-4:~$
```

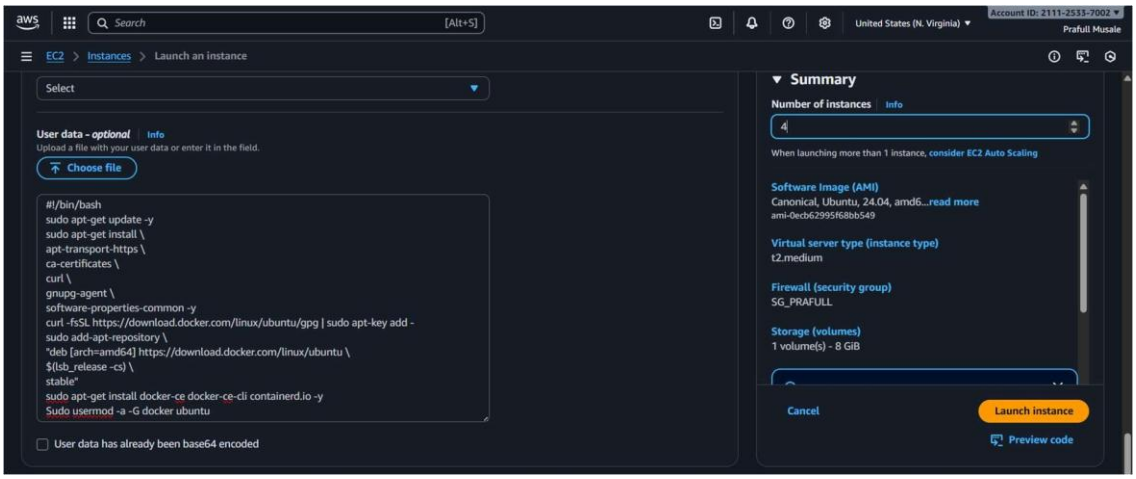
D Y PATIL INTERNATIONAL UNIVERSITY
AKURDI PUNE

School of Computer Science, Engineering and Applications(SCSEA)
B.Tech FIY (CCSA)
Subject : Cloud Automation & Devops (P)

Name of the Student: **Chandrabhushan Lilhare** **PRN: 20220802020**

Title of Practicle : **5.** Web application hosting with docker stack and docker compose (Voting_app)

1. Launch 4 instance with docker installation



2. Connect all instance to terminal

3. Now, Fire $ sudo docker swarm init at 1st instance for give him leadership

```
ubuntu@ip-172-31-24-195:~$ sudo docker swarm init
Swarm initialized: current node (xr2iq9wci3jygl8orvt34m3nx) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-3qcxdb5nvejucsv2csp940enlhhc333wlxzyj35vu54qr0bf0k-7oubd4f7165zxrw417m28b3qj 172.31.24.195:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.
ubuntu@ip-172-31-24-195:~$
```

4. Now generate token for manager and worker with following commands

$ docker swarm join-token manager

$ docker swarm join-token worker

```
ubuntu@ip-172-31-24-195:~$ docker swarm join-token manager
To add a manager to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-3qcxdb5nvejucsv2csp940enlhhc333wlxzyj35vu54qr0bf0k-5y9s
zo95v0z27sbircs2dxp4v 172.31.24.195:2377

ubuntu@ip-172-31-24-195:~$ docker swarm join-token worker
To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-3qcxdb5nvejucsv2csp940enlhhc333wlxzyj35vu54qr0bf0k-7oub
d4f7165zxrw417m28b3qj 172.31.24.195:2377

ubuntu@ip-172-31-24-195:~$
```

5. Now join 2nd instance as a manger with the help of token

Copy manager token and paste into 2nd instance

```
ubuntu@ip-172-31-18-27:~$ docker swarm join --token SWMTKN-1-3qcxdb5nvejucsv2csp940enlhhc333wlxzyj35vu54q
r0bf0k-5y9szo95v0z27sbircs2dxp4v 172.31.24.195:2377
This node joined a swarm as a manager.
ubuntu@ip-172-31-18-27:~$
```

6. Join 3rd and 4th instance as a worker with the help of token

Copy worker token and paste into 3rd and 4th instance

```
ubuntu@ip-172-31-25-4:~$ docker swarm join --token SWMTKN-1-3qcxdb5nvejucsv2csp940enlh
hc333wlxzyj35vu54qr0bf0k-7oubd4f7165zxrw417m28b3qj 172.31.24.195:2377
This node joined a swarm as a worker.
ubuntu@ip-172-31-25-4:~$
```

Name of the Student: **Chandrabhushan Lilhare** **PRN: 20220802020**

Title of Practicle : **5.** Web application hosting with docker stack and docker compose (Voting_app)

```
ubuntu@ip-172-31-20-92:~$ docker swarm join --token SWMTKN-1-3qcxdb5nvejucsv2csp940enl
hhc333wlxzyj35vu54qr0bf0k-7oubd4f7165zxrw417m28b3qj 172.31.24.195:2377
This node joined a swarm as a worker.
ubuntu@ip-172-31-20-92:~$
```

7. Now create docker-stack.yml to write the services

```
version: "3.9"

services:

  redis:
    image: redis:alpine
    networks:
      - frontend

  db:
    image: postgres:15-alpine
    environment:
      POSTGRES_USER: "postgres"
      POSTGRES_PASSWORD: "postgres"
    volumes:
      - db-data:/var/lib/postgresql/data
    networks:
      - backend

  vote:
    image: dockersamples/examplevotingapp_vote
    ports:
      - 5000:80
    networks:
      - frontend
    deploy:
      replicas: 2

  result:
    image: dockersamples/examplevotingapp_result
    ports:
      - 5001:80
    networks:
      - backend

  worker:
    image: dockersamples/examplevotingapp_worker
    networks:
      - frontend
      - backend
    deploy:
      replicas: 2

networks:
  frontend:
  backend:

volumes:
  db-data:
ubuntu@ip-172-31-24-195:~$
```

8. Now deploy with stack with the following command

$ docker stack deploy -c docker-stack.yml votingapp

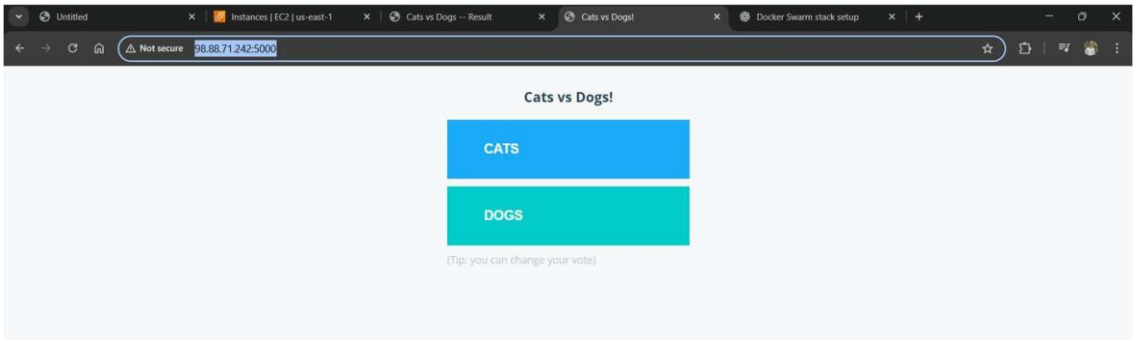
```
ubuntu@ip-172-31-24-195:~$ docker stack deploy -c docker-stack.yml votingapp
Since --detach=false was not specified, tasks will be created in the background.
In a future release, --detach=false will become the default.
Creating network votingapp_backend
Creating network votingapp_frontend
Creating service votingapp_result
Creating service votingapp_worker
Creating service votingapp_redis
Creating service votingapp_db
Creating service votingapp_vote
```

9. Now check stack is created or not with following command

$docker stack ls

```
ubuntu@ip-172-31-24-195:~$ docker stack ls
NAME        SERVICES
votingapp   5
ubuntu@ip-172-31-24-195:~$
```

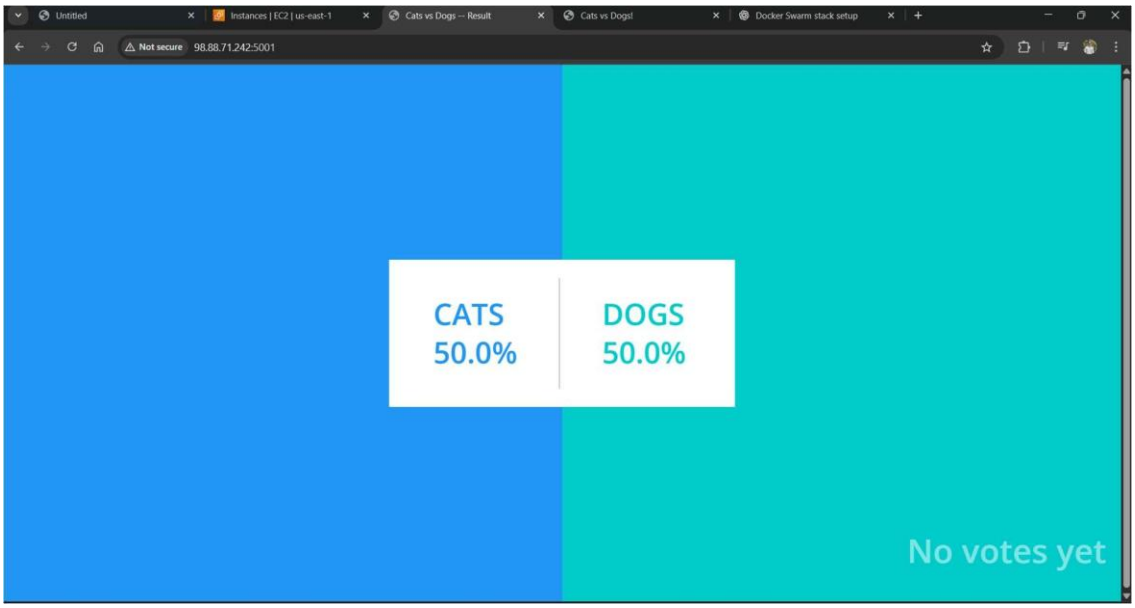
10. Now for voting enter on browser <Public-IP>:5000



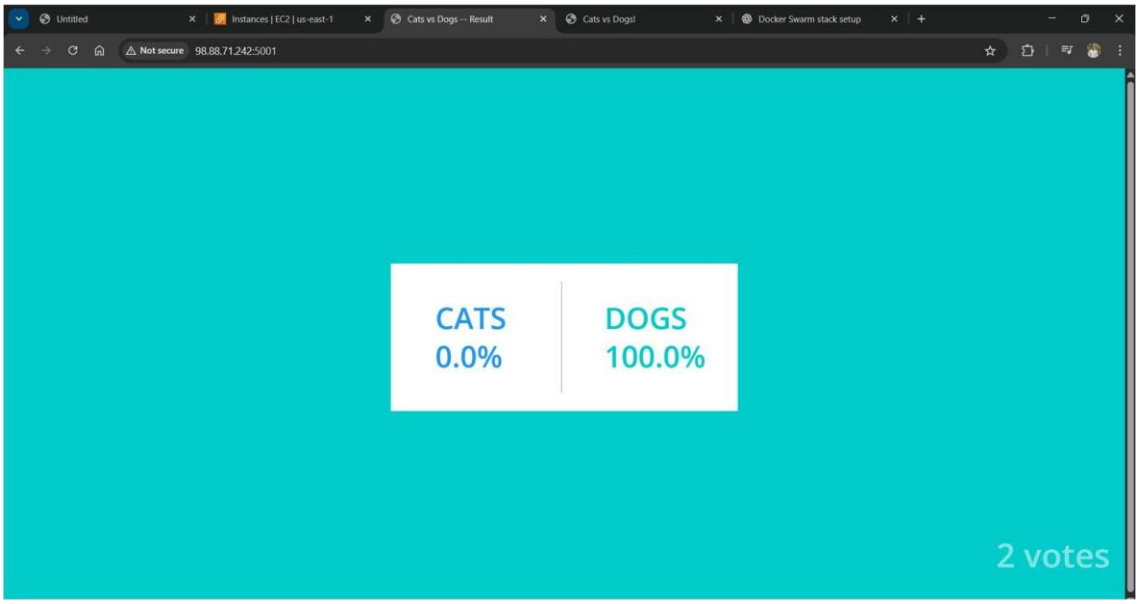
Name of the Student: **Chandrabhushan Lilhare** **PRN: 20220802020**

Title of Practicle : **5.** Web application hosting with docker stack and docker compose (Voting_app)

11. For checking Result enter on browser <Public-IP>:5001



12. Our Voting app is properly working

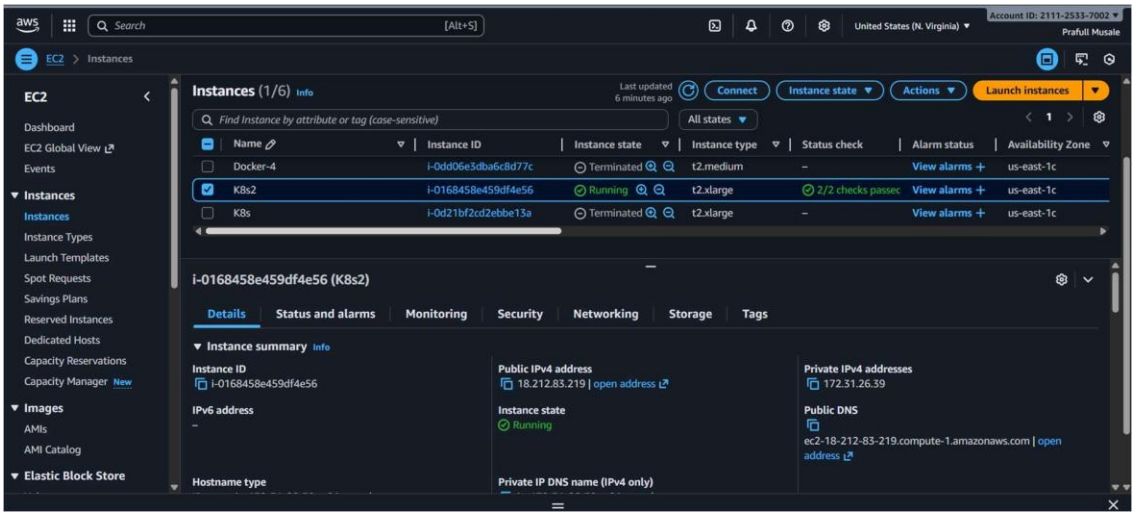


School of Computer Science, Engineering and Applications(SCSEA)
B.Tech FIY (CCSA)
Subject : Cloud Automation & Devops (P)

Name of the Student: **Chandrabhushan Lilhare** **PRN: 20220802020**

Title of Practicle : **6. Deploying multi-container on Kubernetes**

1. Launch an instance with installation of docker and Kubernetes



2. User data for installation

```
#!/bin/bash
sudo apt-get update -y
sudo apt-get install \
apt-transport-https \
ca-certificates \
curl \
gnupg-agent \
software-properties-common -y
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository \
"deb [arch=amd64] https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) \
```

```
stable"

sudo apt-get install docker-ce docker-ce-cli containerd.io -y

sudo usermod -a -G docker ubuntu

 # Initialize Kubernetes cluster

sudo kubeadm init --pod-network-cidr=192.168.0.0/16 --ignore-preflight-
errors=NumCPU || (echo 'Failed to initialize cluster' && exit 1)


# Setup kubectl

mkdir -p $HOME/.kube ; sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config ||
echo 'Failed to copy admin.conf' ; sudo chown $(id -u):$(id -g) $HOME/.kube/config ||
echo 'Failed to change ownership of config'


# Install Calico CNI

kubectl apply -f
https://raw.githubusercontent.com/projectcalico/calico/v3.28.3/manifests/calico.yam
l


# Save join command with full permissions

sudo kubeadm token create --print-join-command
```

Name of the Student: Chandrabhushan Lilhare **PRN: 20220802020**

Title of Practicle : 6. Deploying multi-container on Kubernetes

3. Now create multi-container.yml file

```
apiVersion: v1
kind: Pod
metadata:
  name: two-containers
spec:
  restartPolicy: OnFailure
  initContainers:
    - name: populate-content
      image: debian:stable-slim
      command: ["/bin/sh", "-c"]
      args:
        - mkdir -p /pod-data && echo "Hello from the init container" > /pod-data/index.html
      volumeMounts:
        - name: shared-data
          mountPath: /pod-data
  containers:
    - name: nginx-container
      image: nginx:stable-alpine
      ports:
        - containerPort: 80
      volumeMounts:
        - name: shared-data
          mountPath: /usr/share/nginx/html:ro
  volumes:
    - name: shared-data
      emptyDir: {}

~
~
"multi-container.yml" 27L, 651B                                    27,0-1        All
```

4. Then apply it

```
ubuntu@ip-172-31-26-39:~$ kubectl apply -f multi-container.yml
pod/two-containers created
ubuntu@ip-172-31-26-39:~$ kubectl get pods -w
```

5. Now check the pods

```
NAME             READY   STATUS    RESTARTS   AGE
two-containers   1/1     Running   0          23s

^Cubuntu@ip-172-31-26-39:~kubectl port-forward pod/two-containers 8080:8080
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
^[[A^Cubuntu@ip-172-31-26-39:~$ ^C
```