



**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**



### ***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—

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

## **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 before leaving 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

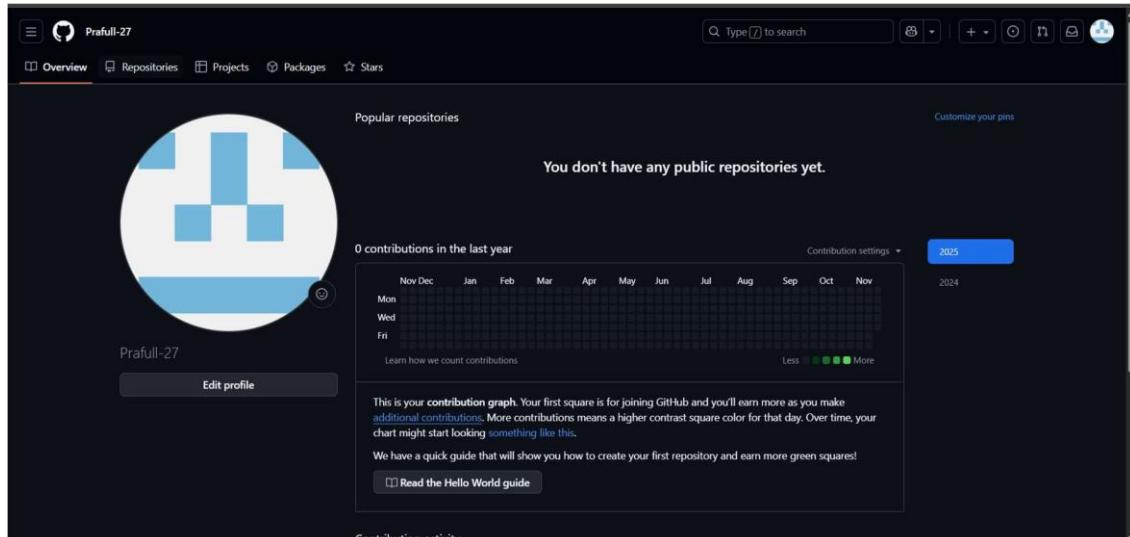


**School of Computer Science, Engineering and Applications(SCSEA)**  
**B.Tech FIY (CCSA)**  
**Subject : Cloud Automation & Devops (P)**

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

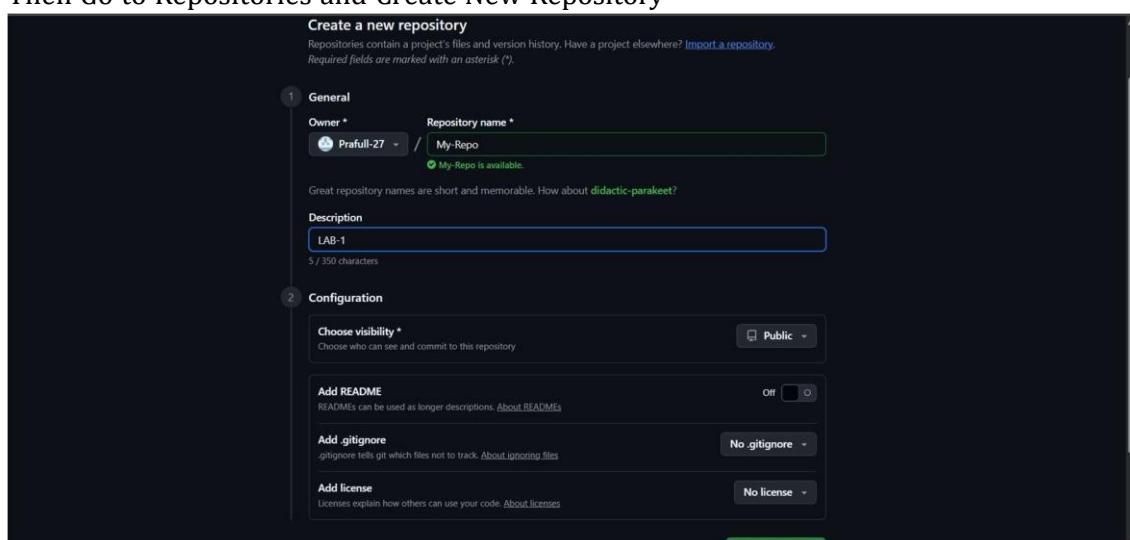
**Title of Practice :** **1.Git workflow with branching, merging, and pull requests**

1. Create account on GitHub



The screenshot shows a GitHub profile for a user named 'Prafull-27'. The profile picture is a light blue circle with a white checkered pattern. The user has 0 contributions in the last year. The GitHub interface is in dark mode.

2. Then Go to Repositories and Create New Repository



The screenshot shows the 'Create a new repository' form on GitHub. The 'General' tab is active, showing the owner as 'Prafull-27' and the repository name as 'My-Repo'. The 'Description' field contains 'LAB-1'. The 'Configuration' tab is partially visible below it, showing options for visibility (set to Public), README, .gitignore, and license.



**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
```

```
ec2-user@ip-172-31-78-23:~$ git clone https://github.com/Prafull-27/My-Repo.git
Cloning into 'My-Repo'...
warning: You appear to have cloned an empty repository.
[ec2-user@ip-172-31-78-23:~]$
```

4. Now our pwd to repository .

```
[ec2-user@ip-172-31-78-23:~]$ cd My-Repo/
[ec2-user@ip-172-31-78-23:My-Repo]$
```

5. Now Create New Branch.

```
[ec2-user@ip-172-31-78-23:My-Repo]$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'
[ec2-user@ip-172-31-78-23:My-Repo]$
```

6. Make changes in your branch:

Edit or create any file.

```
[ec2-user@ip-172-31-78-23:My-Repo]$ echo "Hello git workflow" > file.txt
[ec2-user@ip-172-31-78-23:My-Repo]$ cat file.txt
Hello Git Workflow
[ec2-user@ip-172-31-78-23:My-Repo]$
```

7. Add and commit your changes

```
[ec2-user@ip-172-31-78-23:My-Repo]$ git commit -m "Added file.txt in feature branch"
[feature-branch (root-commit) 376062c] Added file.txt in feature branch
Committer: EC2 Default User <ec2-user@ip-172-31-78-23.ec2.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

git config --global --edit

After doing this, you may fix the identity used for this commit with:

git commit --amend --reset-author

1 file changed, 1 insertion(+)
create mode 100644 file.txt
[ec2-user@ip-172-31-78-23:My-Repo]$
```



**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

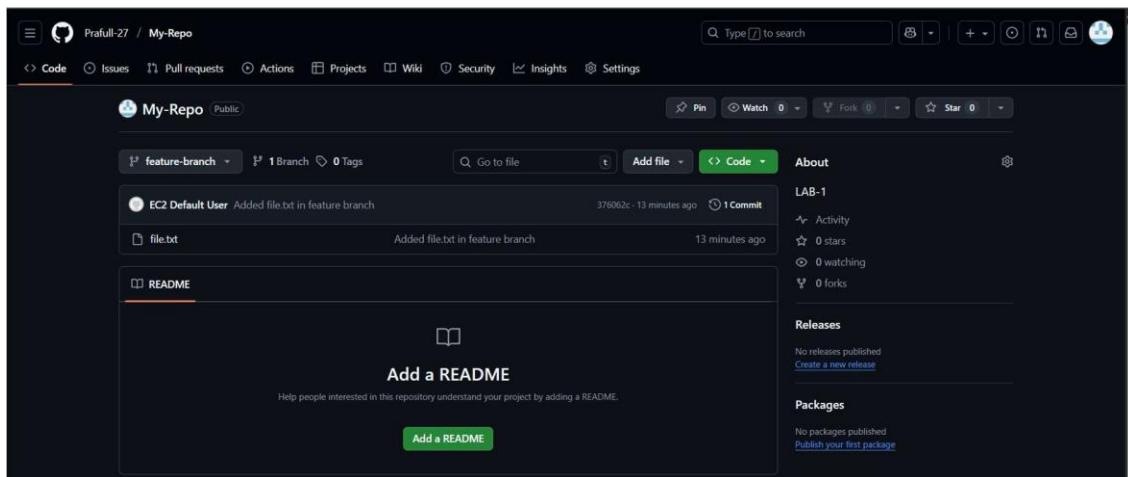
**Title of Practice :** **1.Git workflow with branching, merging, and pull requests**

**8. Push your branch to GitHub**

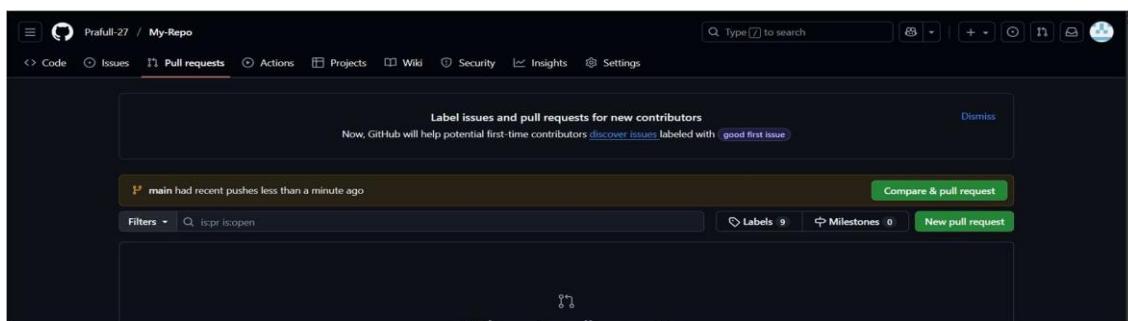
```
[ec2-user@ip-172-31-78-23 My-Repo]$ git push -u origin feature-branch
Username for 'https://github.com': Prafull-27
Password for 'https://Prafull-27@github.com':
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 263 bytes | 263.00 KiB/s, done.
Total 3 (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





**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**

Add a title and description

A screenshot of a GitHub repository titled 'My-Repo'. The user is on the 'Code' tab, specifically the 'Pull requests' section. A new pull request is being created, with the base branch set to 'feature-branch' and the compare branch set to 'main'. The title field contains 'LAB\_1' and the description field contains 'This is LAB 1 of CAD'. The right side of the screen shows various configuration options for the pull request, including reviewers, assignees, labels, projects, milestones, and development details. A note at the bottom indicates that Markdown is supported and files can be added.

Click Create Pull Request

A screenshot of the GitHub pull request creation process. The title and description fields are filled with 'LAB\_1' and 'This is LAB 1 of CAD' respectively. The 'Create pull request' button is visible at the bottom. Below the main form, summary statistics are shown: 1 commit, 1 file changed, and 1 contributor. A note at the bottom left says 'Remember, contributions to this repository should follow our GitHub Community Guidelines.' On the right, there are sections for helpful resources and links to GitHub Community Guidelines.



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

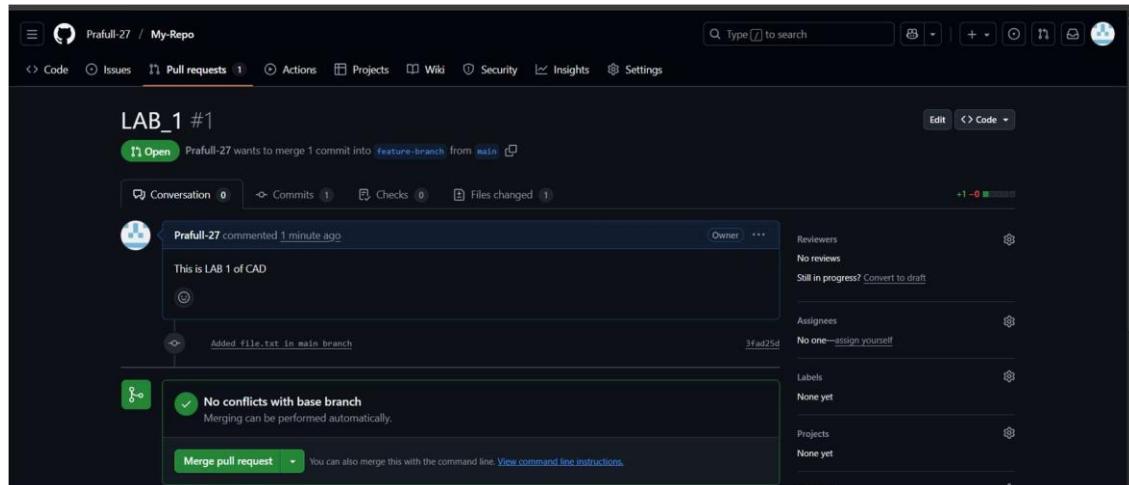
**Name of the Student:** Chandrabhushan Lilhare      **PRN:** 20220802020

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

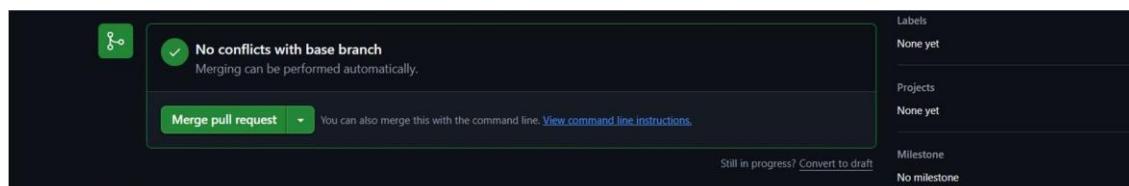
**10. Review and Merge PR**

On GitHub:

1. Open the PR



2. Click **Merge Pull Request**



3. Then click **Confirm Merge**



PRN: 20220802020



**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.

A screenshot of the Docker Hub interface. The top navigation bar shows 'hub', 'Explore', and 'My Hub'. The 'My Hub' tab is selected. On the left, there's a sidebar with options like 'Repositories', 'Hardened Images', 'Collaborations', 'Settings', 'Default privacy', 'Notifications', 'Billing', 'Usage', 'Pulls', and 'Storage'. The main area is titled 'Repositories' with the sub-instruction 'All repositories within the prafull21 namespace.' Below this is a search bar and a dropdown menu. A table lists a single repository: 'prafull21/web\_apps' (Last Pushed: 10 months ago, IMAGE, Public, Inactive). There's also a 'Create a repository' button.

Step 2 — Generate Docker Hub Access Token

1. Go to Account Settings

A screenshot of the Docker Hub interface. The top navigation bar shows 'hub', 'Explore', and 'My Hub'. The 'My Hub' tab is selected. On the right, there's a user profile with a large 'P' icon and the name 'prafull21'. Below the profile are links for 'What's new', 'My profile', 'Account settings' (which is highlighted), and 'Sign out'. The central area features a banner for 'Docker Hardened Images - Secure &amp; Compliant' with a 'Visit catalog now' button. At the bottom, there are links for 'Generative AI' and 'Spotlight'.



**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

**2. Click Person Access Token**

The screenshot shows the Docker Personal access tokens page. The sidebar on the left has a 'Personal access tokens' option selected. The main area displays a table with one row of data:

Description	Scope	Status	Source	Created	Last used	Expiration date
Generated through desktop...	Read, Write, Delete	Active	Auto-generated	Feb 04, 2025 at 20:33:13	Feb 04, 2025 at 20:33:13	Never

**3. Click New Access Token**

The screenshot shows the Docker Personal access tokens page. The sidebar on the left has a 'Personal access tokens' option selected. The main area displays a table with one row of data, identical to the previous screenshot. A blue 'Generate new token' button is visible in the top right corner of the main area.

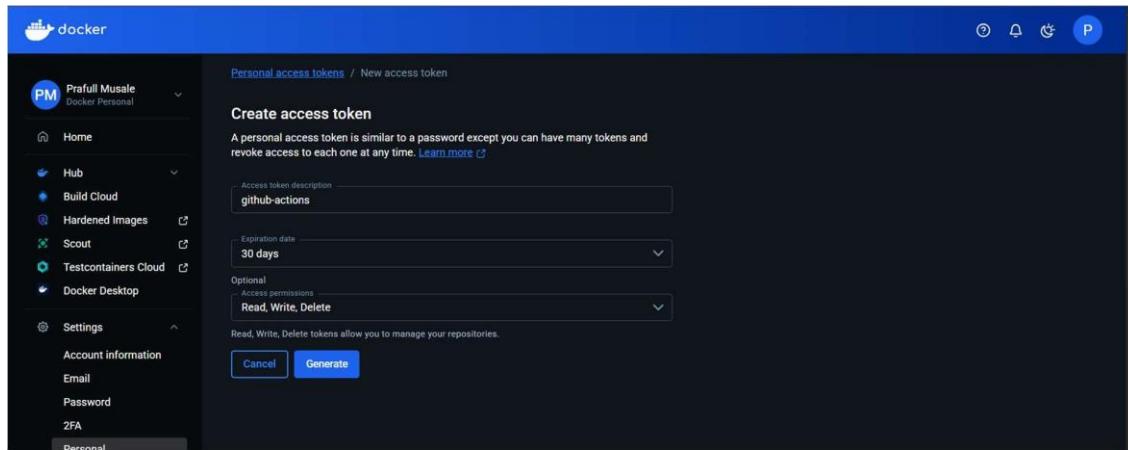


**School of Computer Science, Engineering and Applications(SCSEA)**  
**B.Tech FIY (CCSA)**  
**Subject : Cloud Automation & Devops (P)**

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

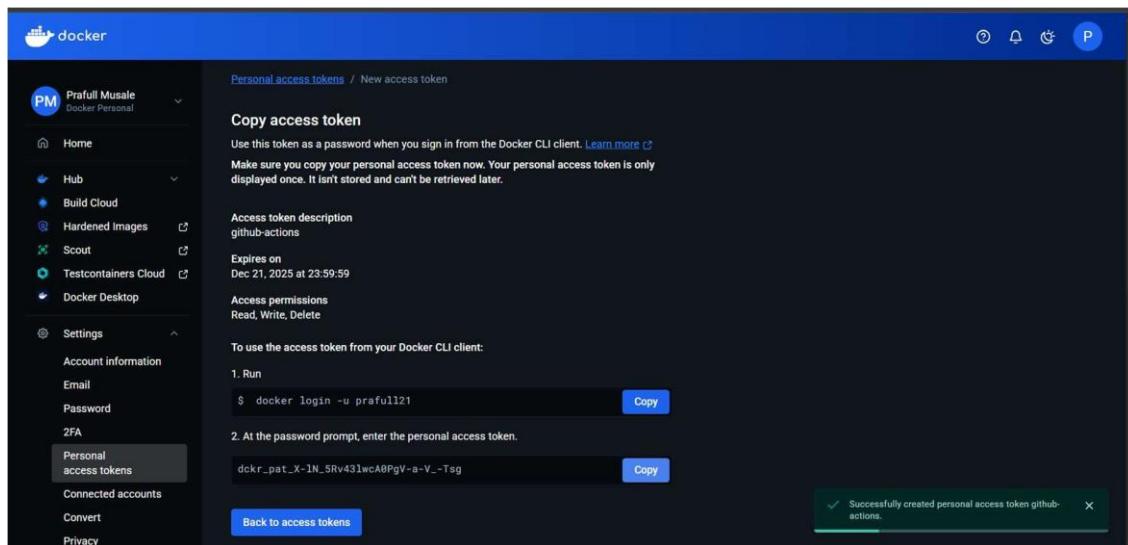
**Title of Practicle :** **2. Setting up a Jenkins CI/CD pipeline for a sample project**

4. Name: github-actions



A screenshot of the Docker Personal access tokens creation page. The left sidebar shows the user's profile (Prafull Musale) and navigation options like Home, Hub, Build Cloud, Hardened Images, Scout, Testcontainers Cloud, Docker Desktop, Settings, Account information, Email, Password, 2FA, and Personal access tokens (which is currently selected). The main content area is titled 'Create access token' and includes fields for 'Access token description' (set to 'github-actions'), 'Expiration date' (set to '30 days'), and 'Optional Access permissions' (set to 'Read, Write, Delete'). Below these fields are two buttons: 'Cancel' and 'Generate'. A note at the bottom states: 'Read, Write, Delete tokens allow you to manage your repositories.'

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



A screenshot of the Docker Personal access tokens copy page. The left sidebar is identical to the previous screen. The main content area is titled 'Copy access token' and displays the copied token: 'dckr\_pst\_X-1N\_5Rv43lwA8PgV-a-V\_-Tsg'. There are two 'Copy' buttons next to the token. Below the token, instructions for using the access token from the Docker CLI client are provided: '1. Run \$ docker login -u prafull21' and '2. At the password prompt, enter the personal access token.' A success message at the bottom right says: 'Successfully created personal access token github-actions.'



**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 3 — Add Secrets in GitHub Repository**

1. Open your repository

The screenshot shows a GitHub repository named 'LAB-2-repo'. The main interface includes sections for 'Start coding with Codespaces', 'Add collaborators to this repository', and a 'Quick setup' guide. The 'Quick setup' section provides instructions for setting up the repository via desktop or command line, mentioning README, LICENSE, and .gitignore files.

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



**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**

The screenshot shows the GitHub repository settings for 'Secrets and variables'. Under 'Environment secrets', it says 'This environment has no secrets.' and has a 'Manage environment secrets' button. Under 'Repository secrets', there are two entries: 'DOCKERHUB\_TOKEN' and 'DOCKERHUB\_USERNAME', both last updated 'now'. The sidebar on the left lists various repository settings like Collaborators, Moderation options, Code and automation, Rules, Actions, Models, Webhooks, Copilot, Environments, Codespaces, Pages, Security, Advanced Security, Deploy keys, and Dependabot.

#### Step 4 — Create Workflow Folder

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

The screenshot shows the GitHub repository interface with a workflow file named 'ci-cd.yml' in the '.github/workflows' folder. The file content is: 'Prafull-27 Create ci-cd.yml'. The repository has 0 lines (0 loc) - 1 Byte. There are buttons for 'Code' and 'Blame'.

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

Inside .github/workflows/, create a file

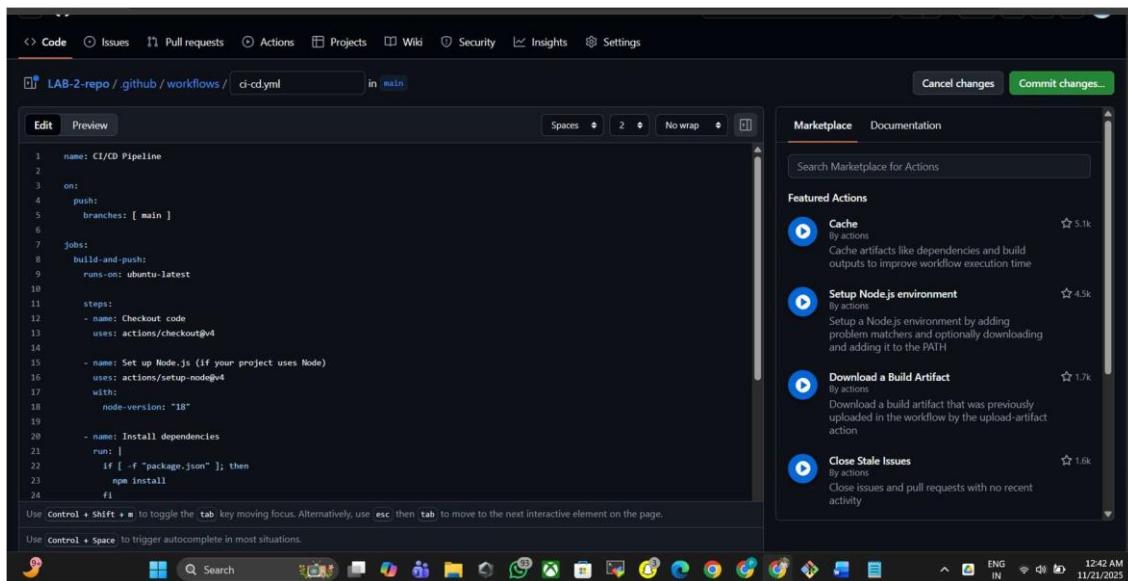
Paste this code inside (simple CI/CD pipeline)



**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



The screenshot shows the GitHub Actions workflow editor for a repository named 'LAB-2-repo'. The workflow file is named 'ci-cd.yml' and contains the following YAML code:

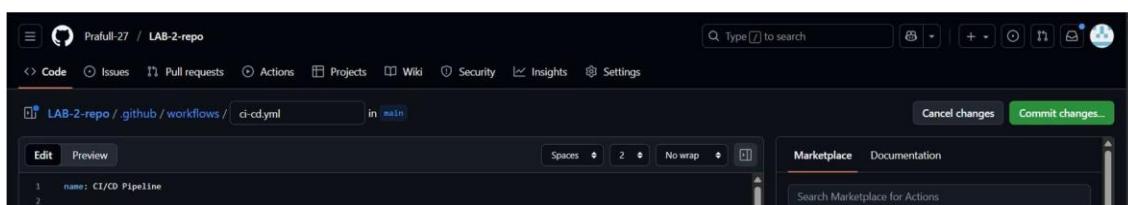
```
name: CI/CD Pipeline
on:
  push:
    branches: [ main ]
jobs:
  build-and-push:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v4
      - name: Set up Node.js (if your project uses Node)
        uses: actions/setup-node@v4
        with:
          node-version: "18"
      - name: Install dependencies
        run:
          if [ -f "package.json" ]; then
            npm install
          fi
```

The right sidebar displays the GitHub Marketplace with several featured actions:

- Cache** By actions Cache artifacts like dependencies and build outputs to improve workflow execution time
- Setup Node.js environment** By actions Setup a Node.js environment by adding problem matchers and optionally downloading and adding it to the PATH
- Download a Build Artifact** By actions Download a build artifact that was previously uploaded in the workflow by the upload-artifact action
- Close Stale Issues** By actions Close issues and pull requests with no recent activity

### Step 6 — Commit & Push

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



The screenshot shows the GitHub Actions workflow editor for the same repository. The workflow file 'ci-cd.yml' now contains only the first two lines:

```
name: CI/CD Pipeline
on:
```

The right sidebar remains the same, showing the GitHub Marketplace with the same four featured actions.



**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 7 — Check the GitHub Actions Pipeline

1. Go to your repository

The screenshot shows a GitHub repository named 'LAB-2-repo'. The main page displays a README file and a '.github/workflows' directory containing two workflow files: 'Add CI/CD pipeline for Docker image build and push' and 'Create ci-cd.yml'. The repository has 3 commits and 0 stars.

2. Click Actions

The screenshot shows the 'Actions' tab in the GitHub repository. It displays a list of workflow runs under the 'All workflows' section. Two runs are listed: 'Add CI/CD pipeline for Docker image build and push' (status: success, 1 minute ago) and 'Create ci-cd.yml' (status: failure, 6 minutes ago).



## 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

3. You will see a new workflow running

The screenshot shows the GitHub Actions interface for a repository named 'LAB-2-repo'. The 'CI/CD Pipeline' tab is selected. On the left, there's a sidebar with options like Management, Caches, Attestations, Runners, Usage metrics, and Performance metrics. The main area displays 10 workflow run results. The first three runs are successful (green checkmarks), while the last one is failed (red X). The runs are listed as follows:

- Add README for sample Node.js application (main) - 3 minutes ago (42s)
- Create index.js (main) - 4 minutes ago (35s)
- Create package.json (main) - 4 minutes ago (46s)
- Add Dockerfile for Node.js application (main) - 5 minutes ago (45s)

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

The screenshot shows the GitHub Actions interface for a repository named 'LAB-2-repo'. A modal dialog titled 'Commit changes' is open over the code editor. The code editor shows a GitHub Actions workflow file named 'ci-cd.yml'. The modal contains fields for 'Commit message' (Add CI/CD pipeline for Docker image build and push) and 'Extended description' (This CI/CD pipeline builds and pushes a Docker image on push to the main branch). Below these fields, there are two radio button options: 'Commit directly to the main branch' (selected) and 'Create a new branch for this commit and start a pull request'. There are also 'Cancel' and 'Commit changes' buttons. To the right of the modal, there's a 'Marketplace' sidebar with various actions like 'Cache', 'Setup Node.js environment', 'Download a Build Artifact', and 'Close Stale Issues'.



**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 8 — Confirm Docker Image in Docker Hub**

Go to Docker Hub → Repositories →

You will see:

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

A screenshot of the Docker Hub interface. At the top, there's a blue header bar with the 'hub' logo, 'Explore', 'My Hub', a search bar ('Search Docker Hub'), and other navigation icons. Below the header, the user profile 'Prafull Musale' is shown, with a 'Community User' badge. The main area shows two repositories under the 'Repositories' tab:

IMAGE	IMAGE
prafull21/sample-app prafull21	prafull21/web_apps prafull21
Pulls: 19 Stars: 0 Last Updated: 12 minutes ago	Pulls: 33 Stars: 0 Last Updated: 10 months ago



**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

Instance summary for i-Offcda7725e8e7203 (Docker) [Info](#)

Updated less than a minute ago

Instance ID	Public IPv4 address	Private IPv4 addresses
i-Offcda7725e8e7203	44.211.31.26   <a href="#">open address</a>	172.31.65.199
IPv6 address	Instance state	Public DNS
-	Running	ec2-44-211-31-26.compute-1.amazonaws.com   <a href="#">open address</a>
Hostname type	Private IP DNS name (IPv4 only)	Elastic IP addresses
IP name: ip-172-31-65-199.ec2.internal	ip-172-31-65-199.ec2.internal	-
Answer private resource DNS name	Instance type	AWS Compute Optimizer finding
IPv4 (A)	t3.micro	<a href="#">Opt-in to AWS Compute Optimizer for recommendations.</a>
Auto-assigned IP address	VPC ID	<a href="#">Learn more</a>
44.211.31.26 [Public IP]	vpc-0fc6827e18a8c0a7c	
IAM Role	Subnet ID	Auto Scaling Group name
-	subnet-0a48b65174f8469ce	-

**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
```



**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

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

```
docker --version
```

```
docker compose version
```

```
docker-compose version
```

```
[ec2-user@ip-172-31-65-199 ~]$ sudo yum update -y
sudo dnf install docker -y
sudo systemctl start docker
sudo systemctl enable docker
sudo usermod -aG docker ec2-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
sudo chmod +x /usr/local/bin/docker-compose
docker --version
docker compose version
docker-compose version|
```

```
Running scriptlet: container-selinux-4:2.242.0-1.amzn2023.noarch
Running scriptlet: docker-25.0.13-1.amzn2023.0.2.x86_64
Verifying : container-selinux-4:2.242.0-1.amzn2023.noarch
Verifying : containerd=2.1.4-1.amzn2023.0.2.x86_64
Verifying : docker-25.0.13-1.amzn2023.0.2.x86_64
Verifying : iptables-libs-1.8.8-3.amzn2023.0.2.x86_64
Verifying : iptables-nft-1.8.8-3.amzn2023.0.2.x86_64
Verifying : libcgroup=3.0-1.amzn2023.0.1.x86_64
Verifying : libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64
Verifying : libnftnl-1.0.1-19.amzn2023.0.2.x86_64
Verifying : libnftnl-1.2.2-2.amzn2023.0.2.x86_64
Verifying : pigz-2.5-1.amzn2023.0.3.x86_64
Verifying : runc-1.3.3-2.amzn2023.0.1.x86_64
11/11
11/11
1/11
2/11
3/11
4/11
5/11
6/11
7/11
8/11
9/11
10/11
11/11

Installed:
  container-selinux-4:2.242.0-1.amzn2023.noarch
  docker-25.0.13-1.amzn2023.0.2.x86_64
  iptables-nft-1.8.8-3.amzn2023.0.2.x86_64
  libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64
  libnftnl-1.2.2-2.amzn2023.0.2.x86_64
  runc-1.3.3-2.amzn2023.0.1.x86_64
  containerd=2.1.4-1.amzn2023.0.2.x86_64
  iptables-libs-1.8.8-3.amzn2023.0.2.x86_64
  libcgroup=3.0-1.amzn2023.0.1.x86_64
  libnftnl-1.0.1-19.amzn2023.0.2.x86_64
  pigz-2.5-1.amzn2023.0.3.x86_64

Complete!
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
% Total    % Received % Xferd  Average Speed   Time   Time Current
          Dload Upload Total Spent   Left Speed
0       0      0      0      0      0      0 --::-- --::-- --::-- 0
100 60.2M 100 60.2M 0     0  166M      0 --::-- --::-- --::-- 166M
Docker version 25.0.13, build 0bab007
docker: 'compose' is not a docker command.
See 'docker --help'.
Docker Compose version v2.29.2
[ec2-user@ip-172-31-65-199 ~]$
```



**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

### 3. Create Your Project Folder

```
mkdir myapp
```

```
cd myapp
```

```
[ec2-user@ip-172-31-65-199:~] $ mkdir myapp
[ec2-user@ip-172-31-65-199:~] $ ls
myapp
[ec2-user@ip-172-31-65-199:~] $ cd myapp/
[ec2-user@ip-172-31-65-199: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"));
```



**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

```
ec2-user@ip-172-31-65-199:~ % + 
const express = require("express");
const app = express();

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

app.listen(3000, "0.0.0.0", () => {
  console.log("Server running on port 3000");
});
```

Create package.json

\$ package.json

```
{
  "name": "docker-app",
  "version": "1.0.0",
  "dependencies": {
    "express": "^4.17.1"
  }
}
```

```
ec2-user@ip-172-31-65-199:~ % + 
{
  "name": "docker-app",
  "version": "1.0.0",
  "dependencies": {
    "express": "^4.17.1"
  }
}|
```



**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

### 5. Write the Dockerfile

Create Dockerfile

```
$ vi dockerfile
```

```
FROM node:18
```

```
WORKDIR /app
```

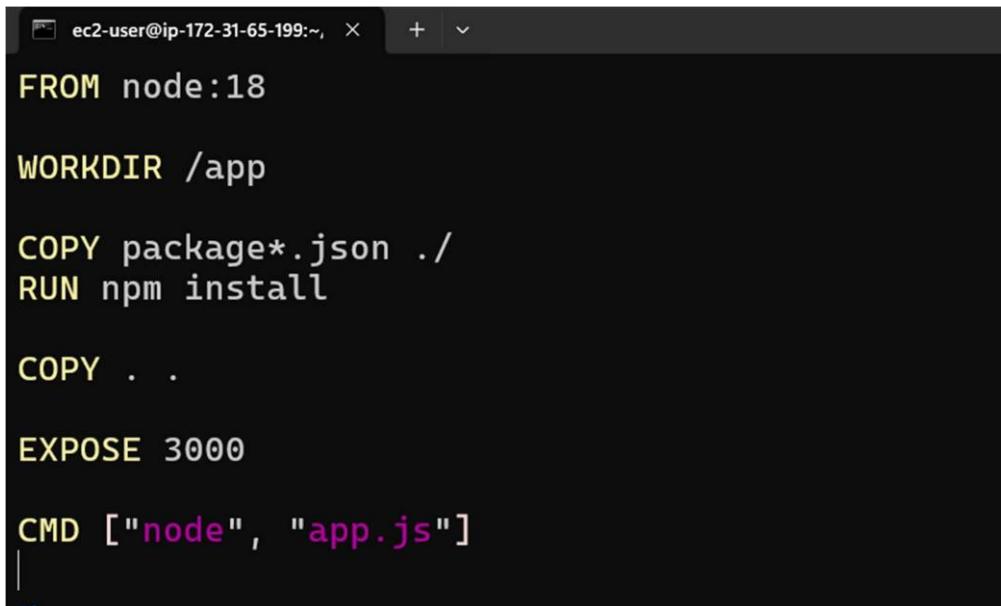
```
COPY package*.json ./
```

```
RUN npm install
```

```
COPY ..
```

```
EXPOSE 3000
```

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

A screenshot of a terminal window titled "ec2-user@ip-172-31-65-199:~". The window contains a Dockerfile with the following content:

```
FROM node:18
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY ..
EXPOSE 3000
CMD ["node", "app.js"]
```

The terminal prompt shows a cursor at the bottom.



**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

## 6. Build Docker Image

\$ docker build -t myapp .

```
[ec2-user@ip-172-31-65-199 myapp]$ docker build -t myapp .
[+] Building 26.8s (10/10) FINISHED
   => [internal] load build definition from dockerfile
   => => transferring dockerfile: 210B                                            docker:default
   => [internal] load metadata for docker.io/library/node:18
   => [internal] load .dockerignore
   => => transferring context: 2B
   => [1/5] FROM docker.io/library/node:18@sha256:c6ae79e38498325db67193d391e6ec1d224d96c693a8a4d94349855 19.9s
   => => resolve docker.io/library/node:18@sha256:c6ae79e38498325db67193d391e6ec1d224d96c693a8a4d94349855 0.0s
   => sha256:eb29363371ee2859fad6a3c5af88d4abc6ff7d399addb13b7de3c1f11bdee6b9 2.49kB / 2.49kB 0.0s
   => sha256:37927ed901b1b2608b72796c6881bf645480268eca4ac9a37b9219e050bb4d84 24.82MB / 24.82MB 0.8s
   => sha256:c6ae79e38498325db67193d391e6ec1d224d96c693a8a4d943498556716d3783 6.41kB / 6.41kB 0.0s
   => sha256:b50082bc3670d0396b2d90e4b0e5bb10265ba5d0ee16bf40f9a505f7045ee563 6.39kB / 6.39kB 0.0s
   => sha256:3e6b9d1a95114e19f12262a4e8a59ad1dia10ca7b82108adcfc0605a200294964 48.49MB / 48.49MB 0.9s
   => sha256:79b2f47ad443652b9b5cc81a95ede249fd976310efdbbe159f29638783778c0 64.40MB / 64.40MB 1.0s
   => => extracting sha256:c6eb9d1a95114e19f12262a4e8a59ad1dia10ca7b82108adcfc0605a200294964 3.2s
   => sha256:e23f099911d692f62b851cf49a1e93294288a115f5cd2d014180e4d3684d34ab 211.36MB / 211.36MB 4.2s
   => sha256:cda7f44f2bddcc4bb7514474024b3f3705de00ddbd6355a33be5ac7808e5b7125 3.32kB / 3.32kB 1.0s
   => sha256:c6b30c3f16966552af10ac00521f60355b1fcfd46ac1c20b1038587e28583ce7 45.68MB / 45.68MB 1.6s
   => sha256:3697be50c98b9d071df4637e1d3491d00e7b9f3a732768c876d82309b3c5a145 1.25MB / 1.25MB 1.1s
   => sha256:461077a72fb7fe40d34a37d6a1958c4d16772d00dd77f572ec50a1fdc41a3754d 446B / 446B 1.2s
   => => extracting sha256:37927ed901b1b2608b72796c6881bf645480268eca4ac9a37b9219e050bb4d84 0.7s
   => => extracting sha256:79b2f47ad443652b9b5cc81a95ede249fd976310efdbbe159f29638783778c0 3.1s
   => => extracting sha256:e23f099911d692f62b851cf49a1e93294288a115f5cd2d014180e4d3684d34ab 8.5s
   => => extracting sha256:cda7f44f2bddcc4bb7514474024b3f3705de00ddbd6355a33be5ac7808e5b7125 0.0s
   => => extracting sha256:c6b30c3f16966552af10ac00521f60355b1fcfd46ac1c20b1038587e28583ce7 2.5s
   => => extracting sha256:3697be50c98b9d071df4637e1d3491d00e7b9f3a732768c876d82309b3c5a145 0.0s
   => => extracting sha256:461077a72fb7fe40d34a37d6a1958c4d16772d0dd77f572ec50a1fdc41a3754d 0.0s
   => [internal] load build context 0.0s
```

**Check image:**

\$ docker images

```
[ec2-user@ip-172-31-65-199 myapp]$ docker images
REPOSITORY      TAG          IMAGE ID      CREATED        SIZE
myapp           latest       11ddef4c9302   45 seconds ago  1.1GB
[ec2-user@ip-172-31-65-199 myapp]$ |
```



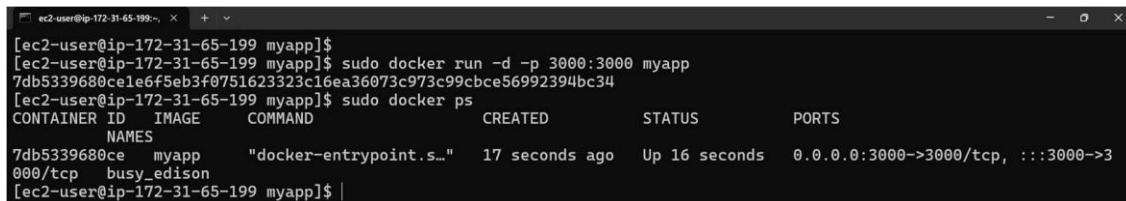
**School of Computer Science, Engineering and Applications(SCSEA)**  
**B.Tech FIY (CCSA)**  
**Subject : Cloud Automation & Devops (P)**

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

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

## 7. Run Container (Without Compose)

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



```
[ec2-user@ip-172-31-65-199 ~]$ [ec2-user@ip-172-31-65-199 myapp]$ sudo docker run -d -p 3000:3000 myapp 7db5339680ce1e6f5eb3f0751623323c16ea36073c973c99cbce56992394bc34 [ec2-user@ip-172-31-65-199 myapp]$ sudo docker ps CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES 7db5339680ce myapp "docker-entrypoint.s..." 17 seconds ago Up 16 seconds 0.0.0.0:3000->3000/tcp, :::3000->3 000/tcp busy_edison [ec2-user@ip-172-31-65-199 myapp]$ |
```

Test:

```
http://<ip>:3000
```



## 8. Create docker-compose.yml

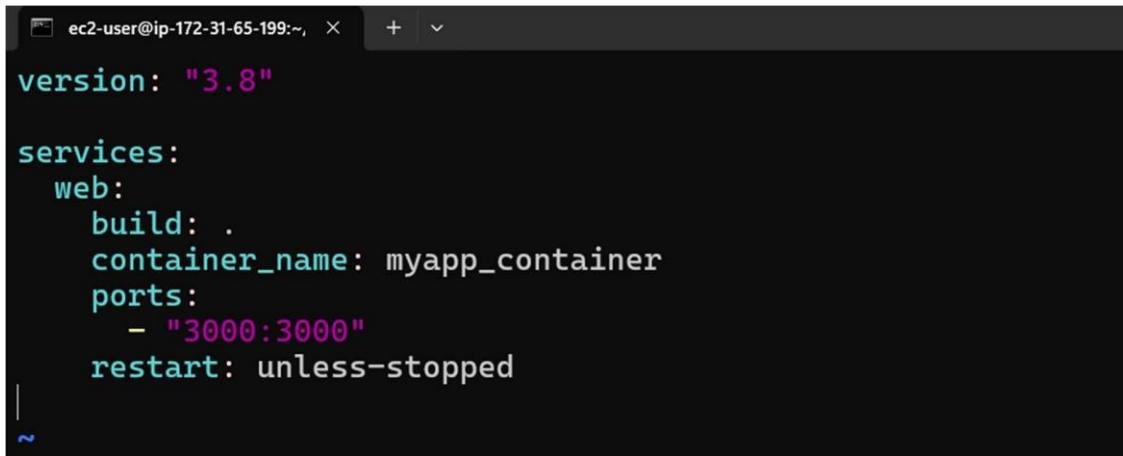
```
$ docker-compose.yml
version: "3.8"
services:
  web:
    build: .
    container_name: myapp_container
    ports:
      - "3000:3000"
    restart: unless-stopped
```



**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



```
ec2-user@ip-172-31-65-199:~ % version: "3.8"

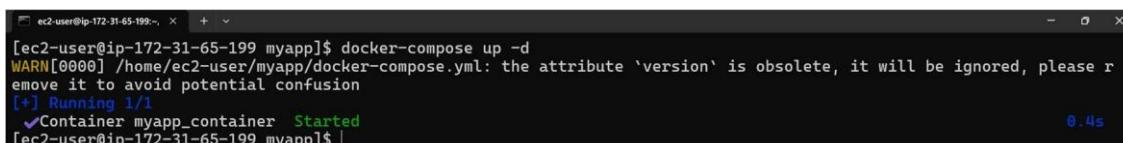
services:
  web:
    build: .
    container_name: myapp_container
    ports:
      - "3000:3000"
    restart: unless-stopped
| ~
```

A screenshot of a terminal window titled 'ec2-user@ip-172-31-65-199:~'. It displays a portion of a Docker Compose configuration file. The file starts with 'version: "3.8"', followed by a 'services' section containing a 'web' service. The 'web' service uses the current directory as its build context ('build: .'), has a container name of 'myapp\_container', maps port 3000 to 3000 ('ports: - "3000:3000"'), and uses the 'unless-stopped' restart policy. A cursor is visible at the end of the file.

## 9. Run Using Docker Compose

**Start:**

```
$ docker-compose up -d
```

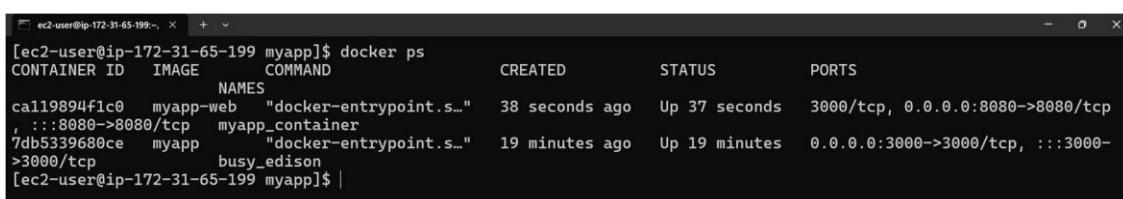


```
[ec2-user@ip-172-31-65-199 myapp]$ docker-compose up -d
WARN[0000] /home/ec2-user/myapp/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] Running 1/1
  ✓ Container myapp_container  Started
[ec2-user@ip-172-31-65-199 myapp]$ | 0.4s
```

A screenshot of a terminal window titled 'ec2-user@ip-172-31-65-199:~'. It shows the command '\$ docker-compose up -d' being run. The output indicates a warning about the 'version' attribute being obsolete and will be ignored. It then shows '[+] Running 1/1' and '[✓ Container myapp\_container Started]', with a duration of '0.4s'.

**Check running:**

```
$ docker ps
```



```
[ec2-user@ip-172-31-65-199 myapp]$ docker ps
CONTAINER ID   IMAGE       COMMAND                  CREATED             STATUS              PORTS
 ca119894f1c0   myapp-web  "docker-entrypoint.s..."  38 seconds ago   Up 37 seconds   3000/tcp, 0.0.0.0:8080->8080/tcp
 , :::8080->8080/tcp   myapp_container
 7db5339680ce   myapp     "docker-entrypoint.s..."  19 minutes ago   Up 19 minutes   0.0.0.0:3000->3000/tcp, :::3000->3000/tcp
   busy_edison
[ec2-user@ip-172-31-65-199 myapp]$ |
```

A screenshot of a terminal window titled 'ec2-user@ip-172-31-65-199:~'. It shows the command '\$ docker ps' being run. The output lists two containers: 'ca119894f1c0' and '7db5339680ce'. Both are in the 'Up' state. The first container is associated with the 'myapp-web' image and has ports 3000/tcp and 0.0.0.0:8080->8080/tcp. The second container is associated with the 'myapp' image and has ports 0.0.0.0:3000->3000/tcp and :::3000->3000/tcp.



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

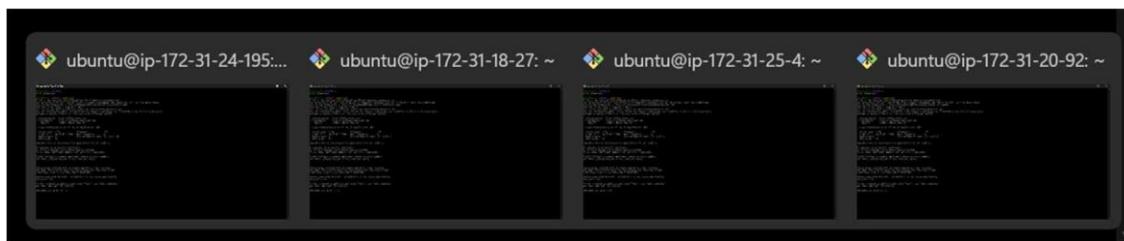
**Name of the Student:** Chandrabhushan Lilhare      **PRN:** 20220802020

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

**1. Launch 4 instance with docker installation**

The screenshot shows the AWS EC2 Instances launch screen. In the 'User data - optional' section, there is a large block of bash script code. The 'Number of instances' dropdown is set to 4. Other settings include Canonical, Ubuntu, 24.04, amd64 as the Software Image (AMI), t2.medium as the Virtual server type, SG\_PRAFULL as the Firewall, and 1 volume(s) - 8 GiB as Storage. The 'Launch instance' button is visible at the bottom right.

**2. Connect all instance to terminal**



**3. Now, Fire \$ sudo docker swarm init at 1<sup>st</sup> instance for give him leadership**

```
ubuntu@ip-172-31-24-195:~$ sudo docker swarm init
Swarm initialized: current node (xr2iq9wci3jyg18orvt34m3nx) 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:~$
```



**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 Practice :** 4. Setup the Docker swarm installation (master slave architecture)

- 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-1-3qcxdb5nvejucsv2csp940en1hhc333wlxzyj35vu54qr0bf0k-5y9s
zo95v0z27sbircs2dpx4v 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-1-3qcxdb5nvejucsv2csp940en1hhc333wlxzyj35vu54qr0bf0k-7oub
d4f7165zxrw417m28b3qj 172.31.24.195:2377

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

- Now join 2<sup>nd</sup> instance as a manager with the help of token

Copy manager token and paste into 2<sup>nd</sup> instance

```
ubuntu@ip-172-31-18-27:~$ docker swarm join --token SWMTKN-1-3qcxdb5nvejucsv2csp940en1hhc333wlxzyj35vu54qr0bf0k-5y9szo95v0z27sbircs2dpx4v 172.31.24.195:2377
This node joined a swarm as a manager.
ubuntu@ip-172-31-18-27:~$ |
```

- Join 3<sup>rd</sup> and 4<sup>th</sup> instance as a worker with the help of token

Copy worker token and paste into 3<sup>rd</sup> and 4<sup>th</sup> instance

```
ubuntu@ip-172-31-25-4:~$ docker swarm join --token SWMTKN-1-3qcxdb5nvejucsv2csp940en1hhc333wlxzyj35vu54qr0bf0k-7oubd4f7165zxrw417m28b3qj 172.31.24.195:2377
This node joined a swarm as a worker.
ubuntu@ip-172-31-25-4:~$ |
```



**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)

```
ubuntu@ip-172-31-20-92:~$ docker swarm join --token SWMTKN-1-3qcxdb5nvejucsv2csp940en1
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
z0fyvuh1hyt06dz676o1y2e8m  ip-172-31-18-27  Ready   Active        Reachable    29.
0.2
aodb8f1fbguax83e4xyxrmmq  ip-172-31-20-92  Ready   Active        Reachable    29.
0.2
xr2iq9wci3jyg18orvt34m3nx * ip-172-31-24-195  Ready   Active        Leader       29.
0.2
atiwsk98cxw05ydhidxuac1s  ip-172-31-25-4   Ready   Active        Reachable    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-1-3qcxdb5nvejucsv2csp940en1h
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 used to view or modify cluster state. Please run this command on a manager node or promote the current node to a manager.
ubuntu@ip-172-31-25-4:~$
```



**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

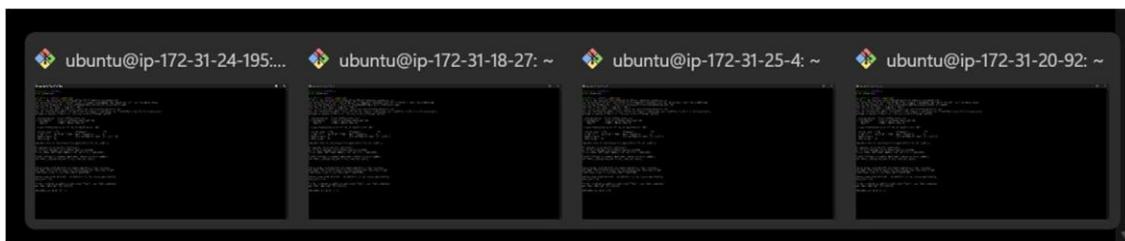
**Title of Practice :** 5. Web application hosting with docker stack and docker compose (Voting\_app)

**1. Launch 4 instance with docker installation**

The screenshot shows the AWS EC2 Instances launch screen. In the 'User data - optional' section, there is a large text area containing a bash script. The script installs Docker, Docker CE, and Containerd, and then installs the Voting app. The 'Number of instances' dropdown is set to 4. Other settings include Canonical, Ubuntu 24.04 AMI, t2.medium instance type, SG\_PRAFULL security group, and 1 volume (8 GiB). The 'Launch instance' button is at the bottom right.

```
#!/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
```

**2. Connect all instance to terminal**



**3. Now, Fire \$ sudo docker swarm init at 1<sup>st</sup> instance for give him leadership**

```
ubuntu@ip-172-31-24-195:~$ sudo docker swarm init
Swarm initialized: current node (xr2iq9wci3jyg18orvt34m3nx) 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:~$
```



**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)

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-1-3qcxdb5nvejucsv2csp940en1hhc333wlxzyj35vu54qr0bf0k-5y9s
zo95v0z27sbircs2dpx4v 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-1-3qcxdb5nvejucsv2csp940en1hhc333wlxzyj35vu54qr0bf0k-7oub
d4f7165zxrw417m28b3qj 172.31.24.195:2377

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

5. Now join 2<sup>nd</sup> instance as a manger with the help of token

Copy manager token and paste into 2<sup>nd</sup> instance

```
ubuntu@ip-172-31-18-27:~$ docker swarm join --token SWMTKN-1-3qcxdb5nvejucsv2csp940en1hhc333wlxzyj35vu54qr0bf0k-5y9szo95v0z27sbircs2dpx4v 172.31.24.195:2377
This node joined a swarm as a manager.
ubuntu@ip-172-31-18-27:~$ |
```

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

Copy worker token and paste into 3<sup>rd</sup> and 4<sup>th</sup> instance

```
ubuntu@ip-172-31-25-4:~$ docker swarm join --token SWMTKN-1-3qcxdb5nvejucsv2csp940en1hhc333wlxzyj35vu54qr0bf0k-7oubd4f7165zxrw417m28b3qj 172.31.24.195:2377
This node joined a swarm as a worker.
ubuntu@ip-172-31-25-4:~$ |
```



**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)

```
ubuntu@ip-172-31-20-92:~$ docker swarm join --token SWMTKN-1-3qcxdb5nvejucsv2csp940en1
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:~$
```



**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 Practice :** 5. Web application hosting with docker stack and docker compose (Voting\_app)

8. Now deploy with stack with the following command

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

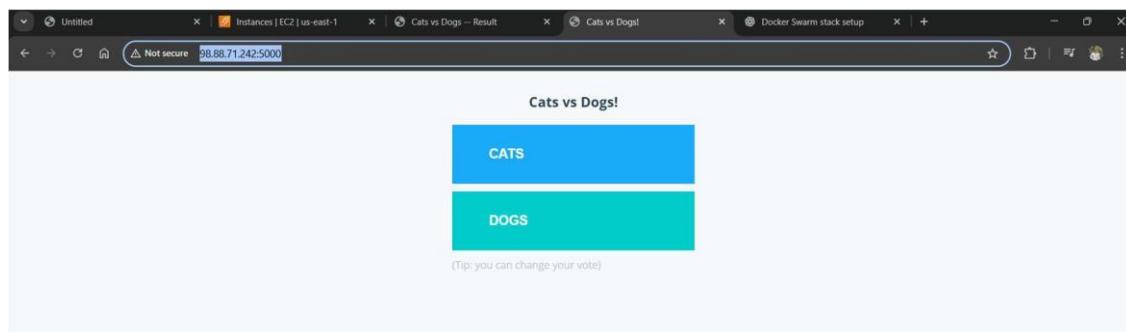
```
ubuntu@ip-172-31-24-195:~$ docker stack deploy -c docker-stack.yml votingapp
Since --detach=false was not specified, tasks will be created in the background.
In a future release, --detach=false will become the default.
Creating network votingapp_backend
Creating network votingapp_frontend
Creating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

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



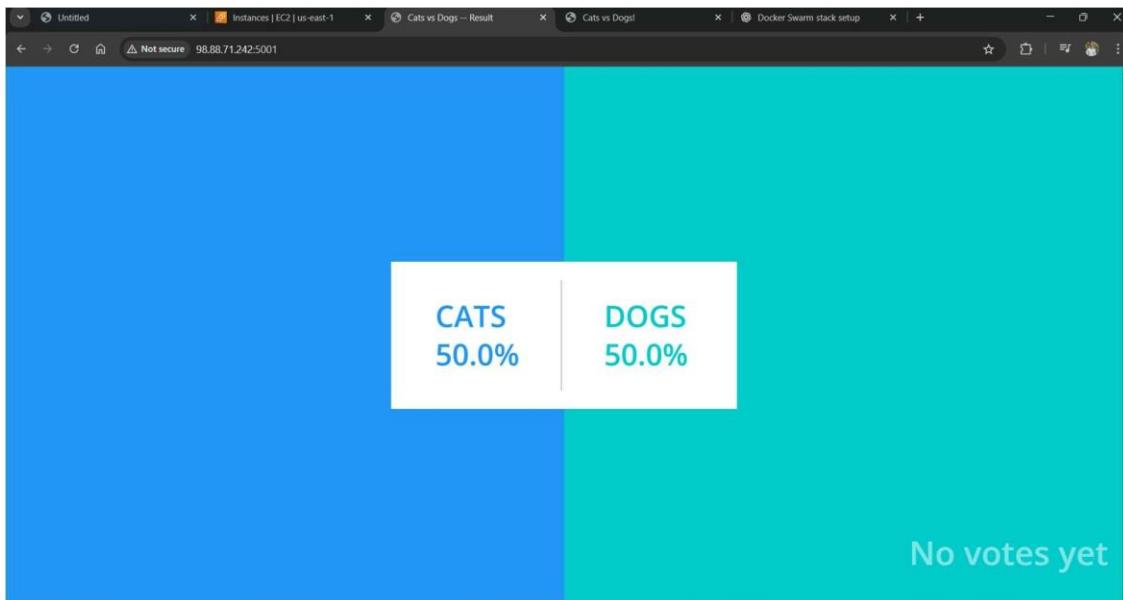


**School of Computer Science, Engineering and Applications(SCSEA)**  
**B.Tech FIY (CCSA)**  
**Subject : Cloud Automation & Devops (P)**

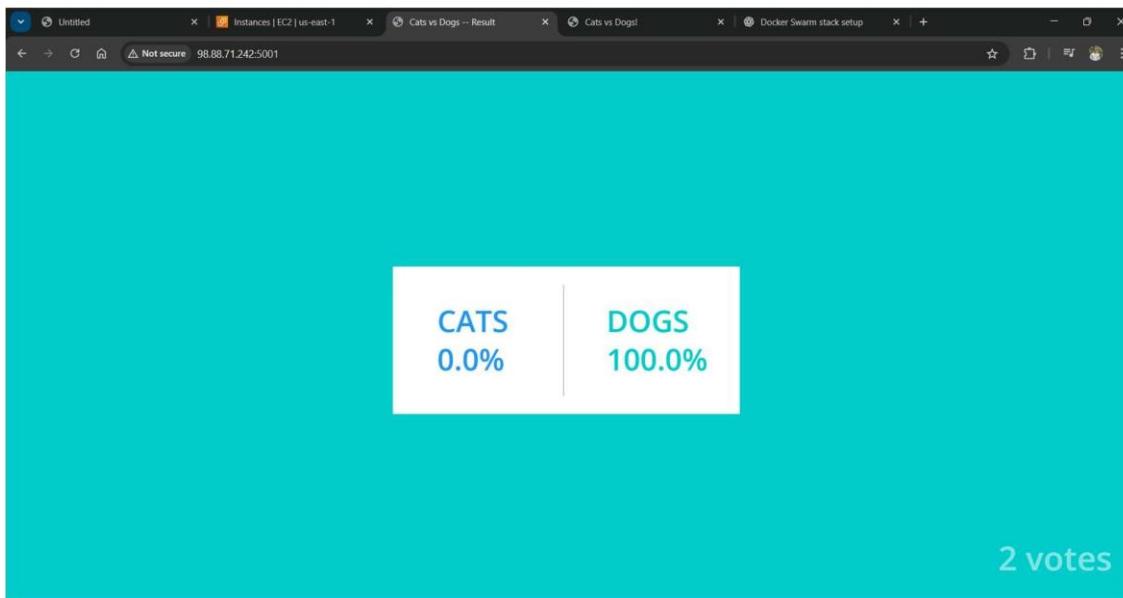
**Name of the Student:** Chandrabhushan Lilhare      **PRN:** 20220802020

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

1. Launch an instance with installation of docker and Kubernetes

The screenshot shows the AWS EC2 Instances page. The left sidebar has 'Instances' selected under 'EC2'. The main table lists three instances:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
Docker-4	i-0dd06e5dba6c8d77c	Terminated	t2.medium	-	<a href="#">View alarms +</a>	us-east-1c
K8s2	i-0168459e459df4e56	Running	t2.xlarge	2/2 checks passed	<a href="#">View alarms +</a>	us-east-1c
K8s	i-0d21bf2cd2ebbe13a	Terminated	t2.xlarge	-	<a href="#">View alarms +</a>	us-east-1c

The details for instance i-0168458e459df4e56 (K8s2) are shown in the details panel:

Details	Status and alarms	Monitoring	Security	Networking	Storage	Tags
<b>Instance summary</b>						
Instance ID i-0168458e459df4e56	Public IPv4 address 18.212.83.219   <a href="#">open address</a>	Instance state Running	Private IPv4 addresses 172.31.26.39			
IPv6 address -	Public DNS ec2-18-212-83-219.compute-1.amazonaws.com   <a href="#">open address</a>	Hostname type -	Private IP DNS name (IPv4 only) -			

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) \
```



**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

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.yaml
```

```
# Save join command with full permissions
```

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



**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

3. Now create multi-container.yml file

```
ubuntu@ip-172-31-26-39:~$ cat multi-container.yml
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

```
ubuntu@ip-172-31-26-39:~$ kubectl get pods -w
NAME          READY   STATUS    RESTARTS   AGE
two-containers 1/1     Running   0          23s

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