

# Task 1: Automate Code Deployment Using CI/CD Pipeline

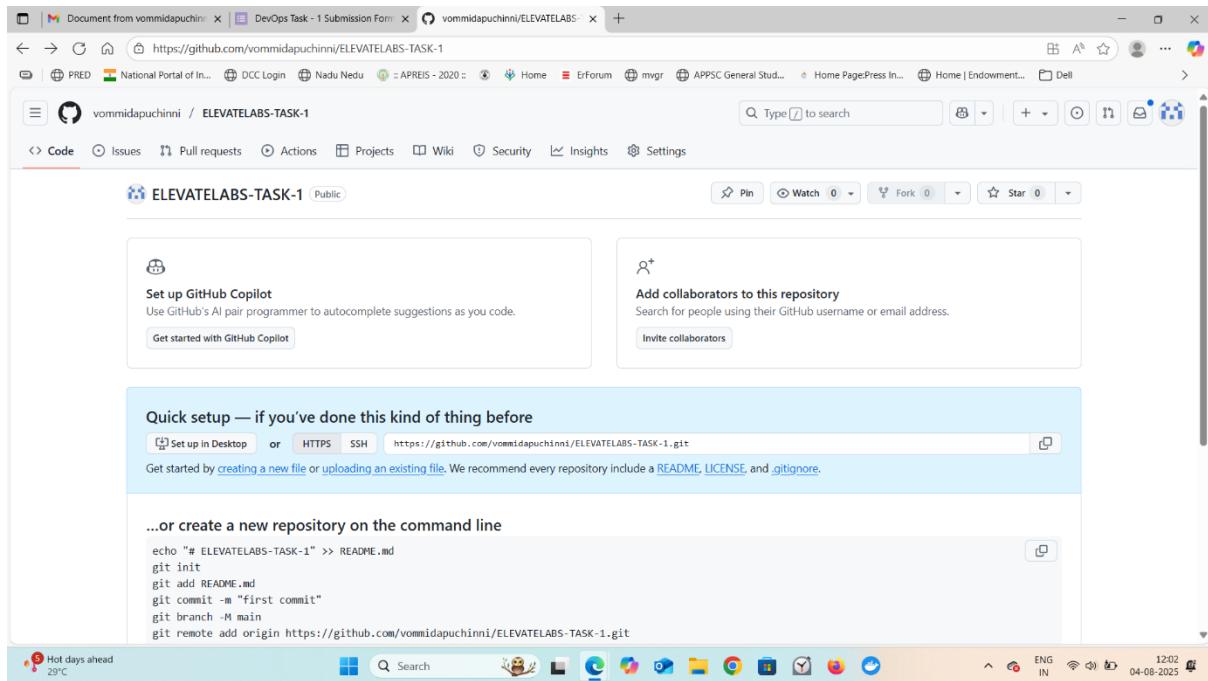
**Objective:** Set up a CI/CD pipeline using GitHub Actions to build and deploy a sample Node.js web app using DockerHub.

## Tools Used

- GitHub → Version control and CI/CD triggering
- GitHub Actions → Workflow automation tool for CI/CD
- Node.js → Backend JavaScript runtime for the sample app
- Docker → Containerization of the application
- DockerHub → Hosting and storing Docker images

## Project Setup:

### Created GitHub Repository



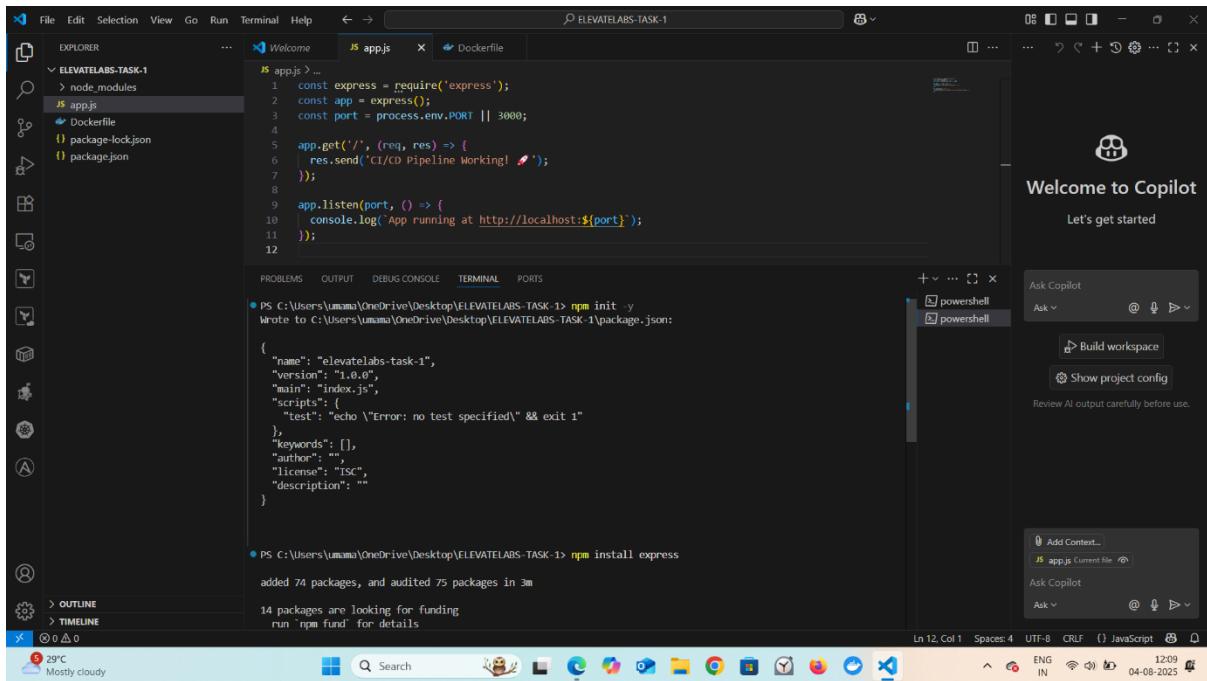
### Install nodejs locally

#### Node.js App Initialization:

```
npm init -y
```

```
npm install express
```

app.js file is created

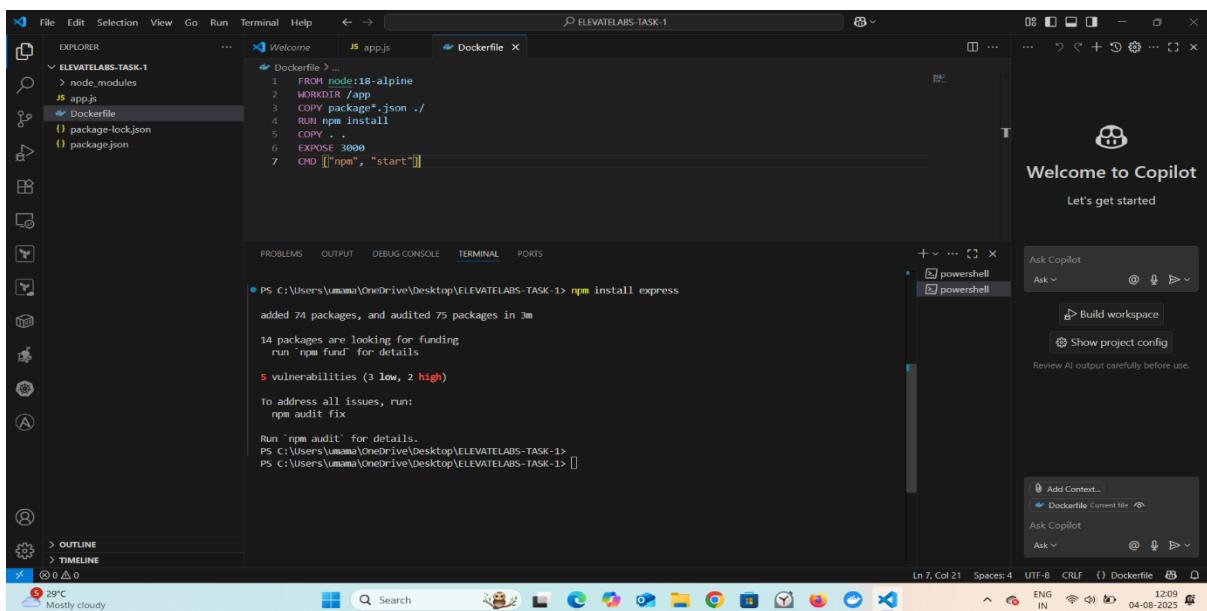


The screenshot shows the VS Code interface with the following details:

- Explorer View:** Shows the project structure with files: Dockerfile, package-lock.json, package.json, and app.js.
- Terminal:** Displays the command `npm init -y` run in the root directory, which generates a package.json file with basic configuration.
- Code Editor:** Shows the app.js file content:1 const express = require('express');
2 const app = express();
3 const port = process.env.PORT || 3000;
4
5 app.get('/', (req, res) => {
6 res.send(`CI/CD Pipeline Working! 🚀`);
7 }
8
9 app.listen(port, () => {
10 console.log(`App running at http://localhost:\${port}`);
11 });
12
- Output Panel:** Shows the result of the `npm install express` command, indicating 75 packages installed and 14 packages looking for funding.
- Right Sidebar:** Features the "Welcome to Copilot" AI integration.

- It's the main entry point for the Node.js application.
- Uses the **Express** module to create a web server.
- The server listens on **port 3000** (or an environment-defined port).
- Defines a route / that responds with: "**CI/CD Pipeline Working!** 🚀".
- Confirms the app is successfully running after deployment.
- Helpful to test if the CI/CD pipeline and Docker image deployment are working as expected.

Docker file created.



The screenshot shows the VS Code interface with the following details:

- Explorer View:** Shows the project structure with files: Dockerfile, package-lock.json, package.json, and app.js.
- Terminal:** Displays the command `npm install express` run in the root directory, showing the same output as before: 75 packages installed and 14 packages looking for funding.
- Code Editor:** Shows the Dockerfile content:FROM node:18-alpine
WORKDIR /app
COPY package\*.json .
RUN npm install
COPY .
EXPOSE 3000
CMD ["npm", "start"]
- Output Panel:** Shows the result of the `npm audit` command, identifying 5 vulnerabilities (3 low, 2 high) and providing instructions to fix them.
- Right Sidebar:** Features the "Welcome to Copilot" AI integration.

- FROM node:14: Uses Node.js base image
- WORKDIR: Sets working directory
- COPY & RUN: Installs dependencies
- CMD: Starts the server

## package.json (script added for testing)

```
"scripts": {
  "start": "node app.js",
  "test": "echo \"No tests yet\" && exit 0"
}
```

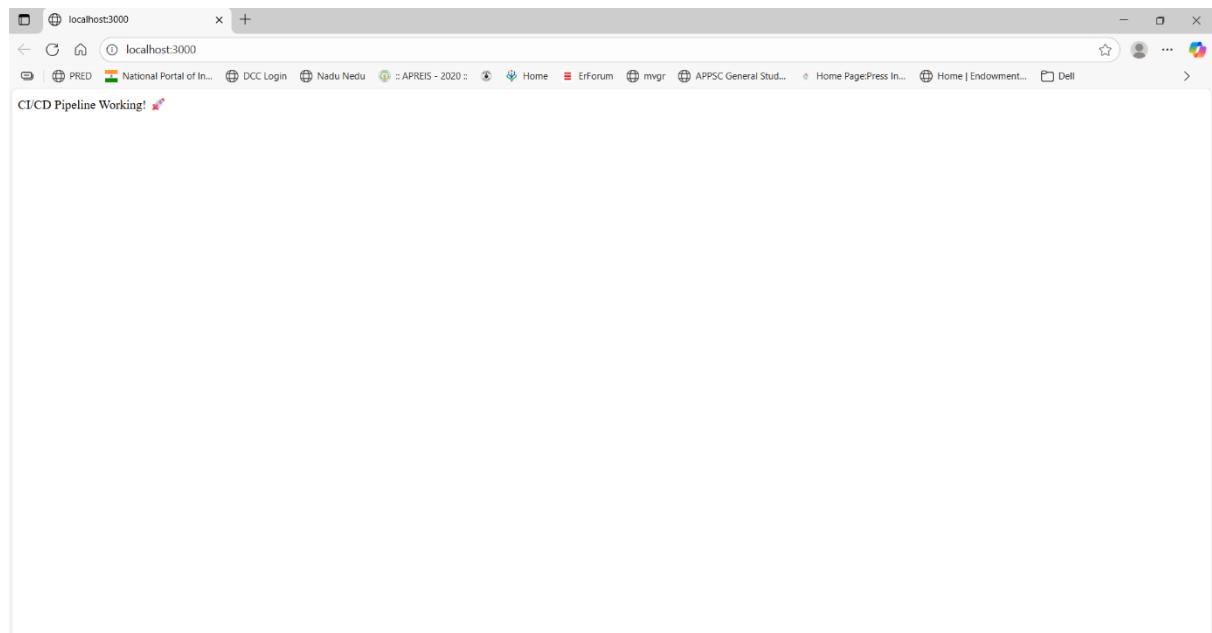
```
PS C:\Users\umama\OneDrive\Desktop\ELEVATELABS-TASK-1> npm install
up to date, audited 75 packages in 6s
14 packages are looking for funding
  run 'npm fund' for details
5 vulnerabilities (3 low, 2 high)

To address all issues, run:
  npm audit fix

Run 'npm audit' for details.
PS C:\Users\umama\OneDrive\Desktop\ELEVATELABS-TASK-1>
PS C:\Users\umama\OneDrive\Desktop\ELEVATELABS-TASK-1> node app.js
App running at http://localhost:3000
```

node app.js will start the server we can access the server by localhost:3000

to stop it we use **ctrl+c**



Create the CI/CD

```

name: CI/CD Pipeline
on:
  push:
    branches: [main]
jobs:
  build-and-deploy:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v3
      - name: Set up Node.js
        uses: actions/setup-node@v3
        with:
          node-version: '18'
      - name: Install dependencies
        run: npm install
      - name: Run tests
        run: npm test
      - name: Login to DockerHub
        run: echo "${{ secrets.DOCKER_PASSWORD }}" | docker login -u "${{ secrets.DOCKER_USERNAME }}" --password-stdin
      - name: Build Docker image
        run: docker build -t ${{ secrets.DOCKER_USERNAME }}/nodejs-demo-app:latest .
      - name: Push Docker image
        run: docker push ${{ secrets.DOCKER_USERNAME }}/nodejs-demo-app:latest

```

This pipeline automatically builds, tests, and pushes Docker image to DockerHub on every push to main.

## Secrets Used:

- DOCKER\_USERNAME – My DockerHub username
- DOCKER\_PASSWORD – My DockerHub password (stored securely in GitHub Secrets)

Name	Last updated
DOCKER_PASSWORD	1 minute ago
DOCKER_USERNAME	now

## Push to GitHub

In terminal run these commands:

git init

- git remote add origin https://github.com/vommidapuchinni/ELEVATELABS-TASK-1.git
  - git add .
  - git commit -m "Initial commit with CI/CD"

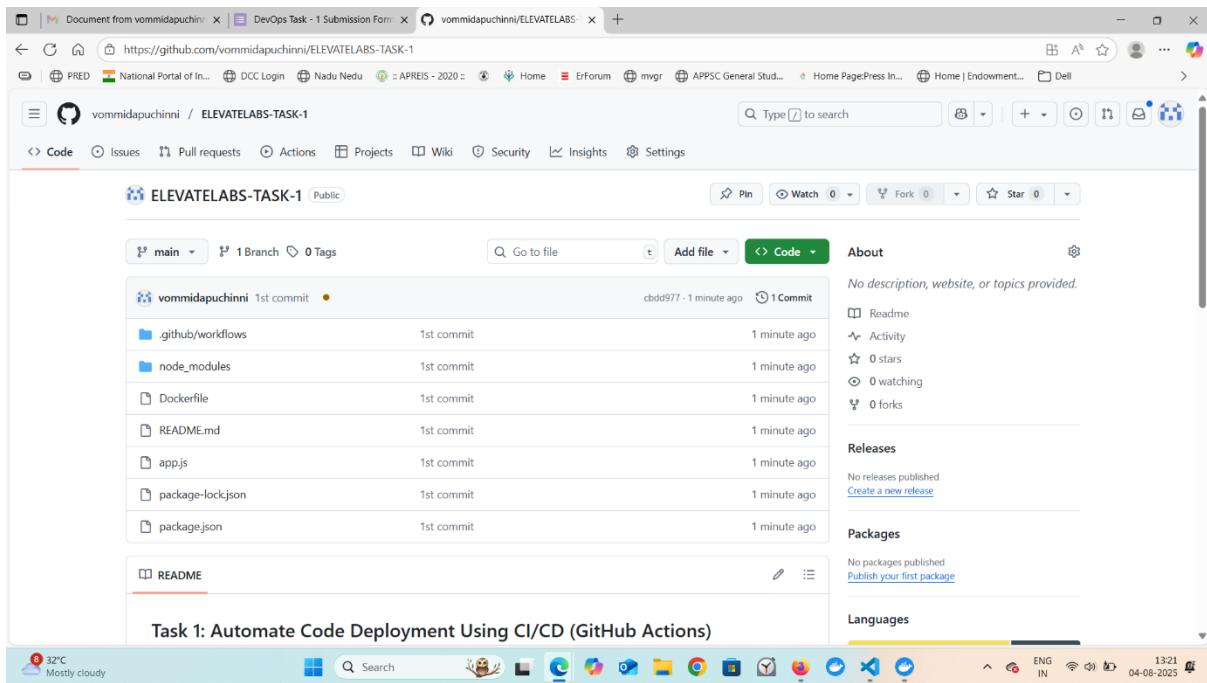
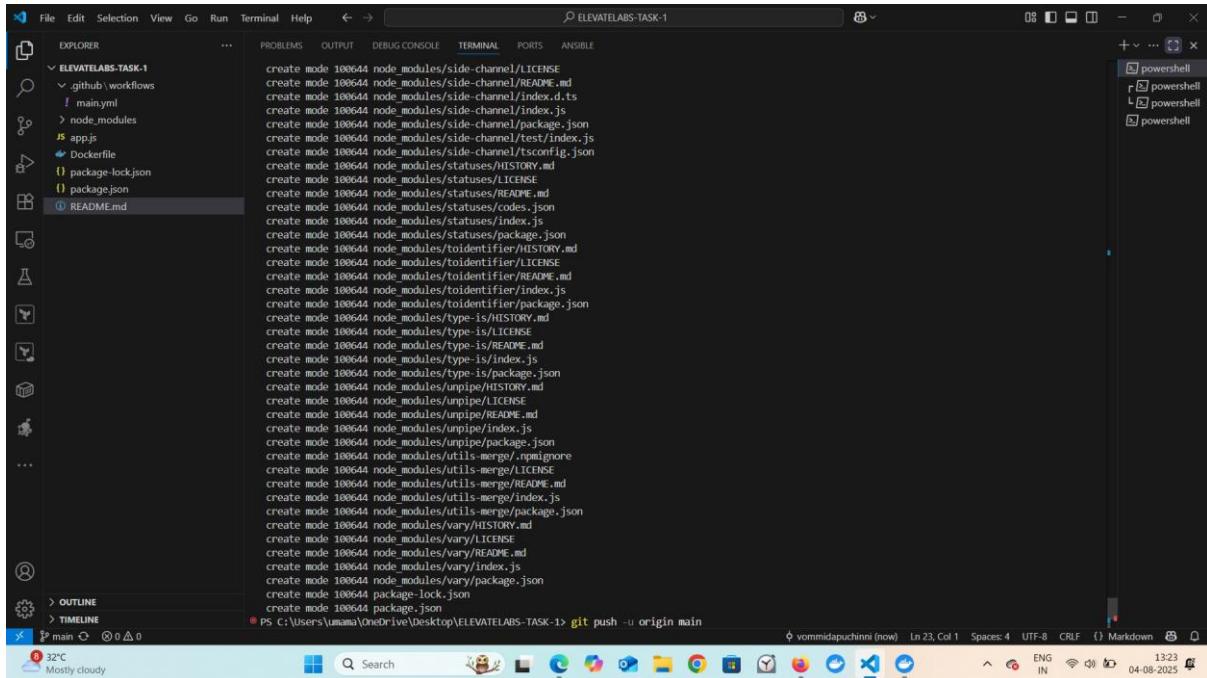
The screenshot shows a Windows desktop environment with the Visual Studio Code application open. The title bar reads "ELEVATELABS-TASK-1". The main area displays a terminal window with the following command and output:

```
PS C:\Users\umama\OneDrive\Desktop\ELEVATELABS-TASK-1> git commit -m "1st commit"
[master (root-commit) cdd997] 1st commit
 669 files changed, 76365 insertions(+)
create mode 100644 .github/workflows/main.yml
create mode 100644 Dockerfile
create mode 100644 README.md
create mode 100644 app.js
create mode 100644 node_modules/.bin/mime
create mode 100644 node_modules/.bin/mime.cmd
create mode 100644 node_modules/.bin/mime.ps1
create mode 100644 node_modules/.package-lock.json
create mode 100644 node_modules/.npmrc/HISTORY.md
create mode 100644 node_modules/.accepts/LICENSE
create mode 100644 node_modules/.accepts/README.md
create mode 100644 node_modules/.accepts/index.js
create mode 100644 node_modules/.accepts/package.json
create mode 100644 node_modules/array-flatten/LICENSE
create mode 100644 node_modules/array-flatten/README.md
create mode 100644 node_modules/array-flatten/array-flatten.js
create mode 100644 node_modules/array-flatten/package.json
create mode 100644 node_modules/body-parser/HISTORY.md
create mode 100644 node_modules/body-parser/LICENSE
create mode 100644 node_modules/body-parser/README.md
create mode 100644 node_modules/body-parser/SECURITY.md
create mode 100644 node_modules/body-parser/index.js
create mode 100644 node_modules/body-parser/lib/read.js
create mode 100644 node_modules/body-parser/lib/types/json.js
create mode 100644 node_modules/body-parser/lib/types/raw.js
create mode 100644 node_modules/body-parser/lib/types/text.js
create mode 100644 node_modules/body-parser/lib/types/urlencoded.js
create mode 100644 node_modules/body-parser/node_modules/qs/.editorconfig
create mode 100644 node_modules/body-parser/node_modules/qs/eslintrc
create mode 100644 node_modules/body-parser/node_modules/qs/.github/FUNDING.yml
create mode 100644 node_modules/body-parser/node_modules/qs/.nycrc
create mode 100644 node_modules/body-parser/node_modules/qs/CHANGELOG.md
create mode 100644 node_modules/body-parser/node_modules/qs/LICENSE.md
create mode 100644 node_modules/body-parser/node_modules/qs/README.md
create mode 100644 node_modules/body-parser/node_modules/qs/dist/qsj
create mode 100644 node_modules/body-parser/node_modules/qslib/formats.js
create mode 100644 node_modules/body-parser/node_modules/qslib/index.js
create mode 100644 node_modules/body-parser/node_modules/qslib/parse.js
```

The Explorer sidebar on the left shows the project structure with files like ".main.yml", "Dockerfile", "app.js", "README.md", and various configuration and package files. The status bar at the bottom shows the date as "04-08-2025".

```
git branch -M main
```

```
git push -u origin main
```

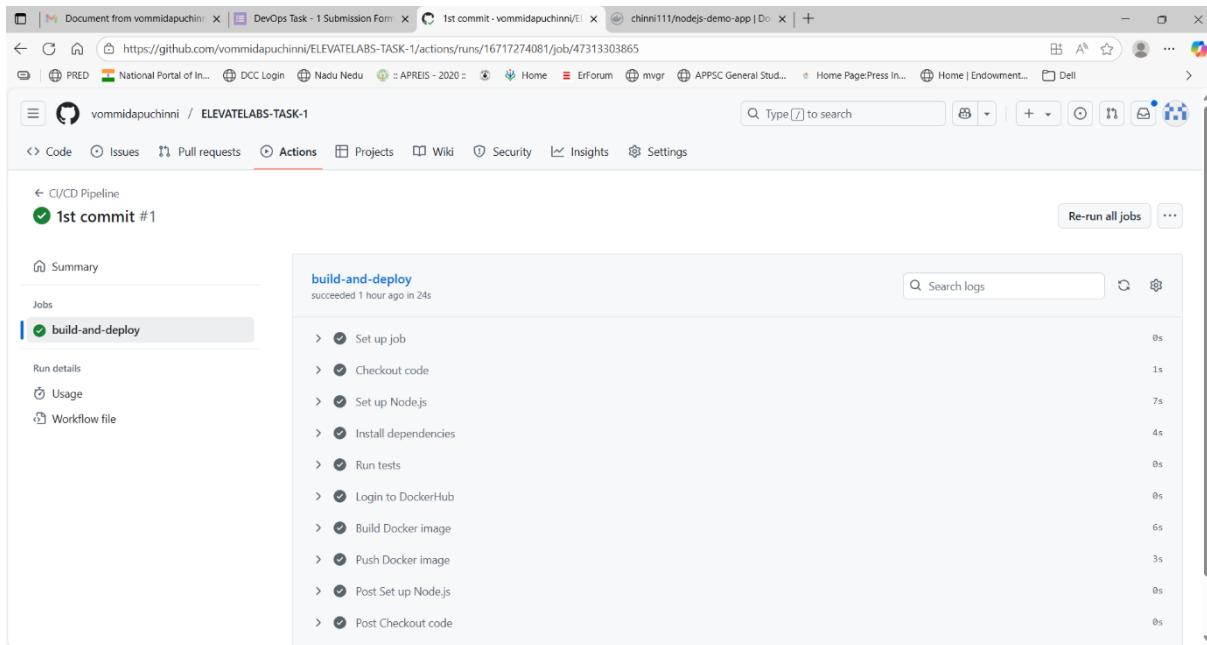


## Project Setup Summary

- Created GitHub repository: nodejs-demo-app
- Wrote app.js to serve a simple route using Express
- Added Dockerfile to containerize the app

- Created GitHub Actions workflow .github/workflows/main.yml to:
  - Trigger on push to main
  - Build Docker image
  - Push to DockerHub

Successfully build and deployed



## CI/CD Workflow Steps

- GitHub push triggers the pipeline
- Pipeline builds the Docker image
- Docker image is pushed to DockerHub
- Pulled the image manually using Docker CLI and verified app running on localhost:3000

## DockerHub:

After successful GitHub Actions run:

- The image was pushed to my DockerHub: chinni111/nodejs-demo-app:latest

The screenshot shows the DockerHub interface for the repository `chinni111/nodejs-demo-app`. The repository was last pushed 21 minutes ago. There is one tag listed: `latest`, which is an `Image` type and was pushed less than 1 day ago. The sidebar on the right features a `buildcloud` integration, which is associated with Docker Build Cloud. It mentions accelerating image build times with cloud-based builders and shared cache. It also highlights Docker Build Cloud's ability to execute builds on optimally-dimensioned cloud infrastructure with dedicated per-organization isolation.

## Running the App Locally:

I did not build the image locally.

Instead, I pulled the image from DockerHub and ran it using:

`docker pull chinni111/nodejs-demo-app`

`docker run -d -p 3000:3000 chinni111/nodejs-demo-app`

Then I accessed the app in my browser using:

<http://localhost:3000>

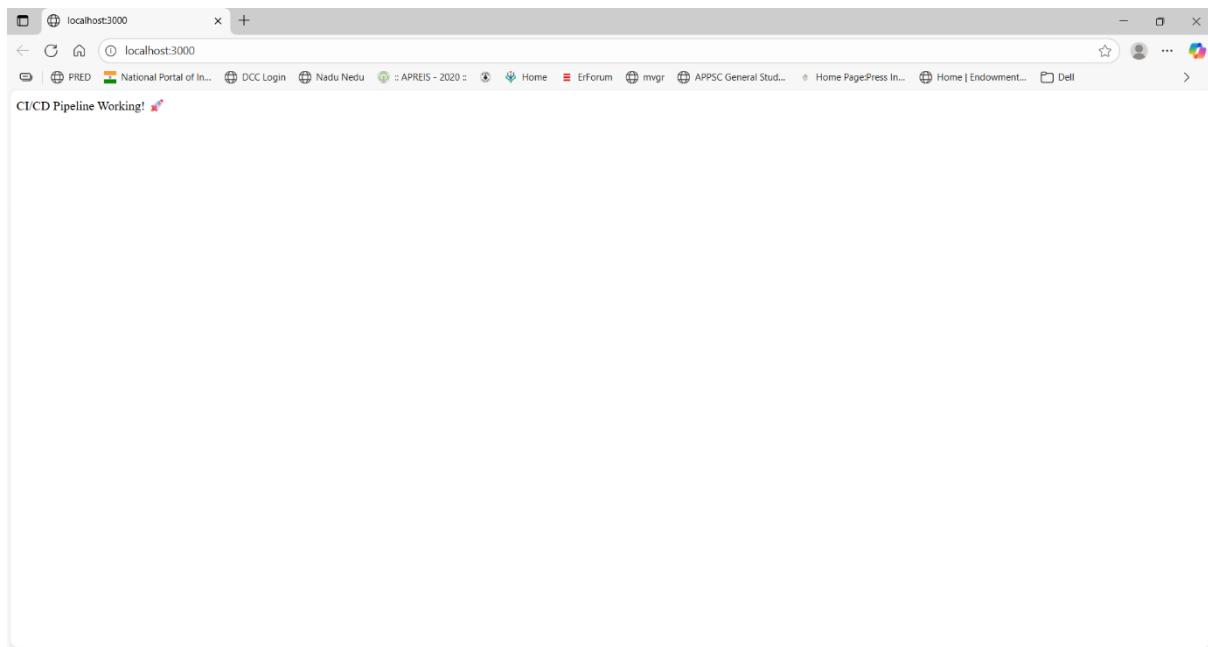
```

PS C:\Users\umama\OneDrive\Desktop\ELEVATELABS-TASK-1> docker pull chinni111/nodejs-demo-app
Using default tag: latest
latest: Pulling from chinni111/nodejs-demo-app
f1823217ab9c: Pull complete
d71dde834b5: Pull complete
1e5a4c90ce05: Pull complete
25ff2da83e41: Pull complete
076c47fb6bb0: Pull complete
724e4d3161cd: Pull complete
92128510276c: Pull complete
29da365019fe: Pull complete
Digest: sha256:bacd3c5e1261f57edce8737f0f1fc93ec4be3b1c46cabc8a1421429b2ad96b69
Status: Downloaded newer image for chinni111/nodejs-demo-app:latest
docker.io/chinni111/nodejs-demo-app:latest

What's next:
View a summary of image vulnerabilities and recommendations → docker scout quickview chinni111/nodejs-demo-app
PS C:\Users\umama\OneDrive\Desktop\ELEVATELABS-TASK-1> docker run -d -p 3000:3000 chinni111/nodejs-demo-app
2d1ae4cd9c07: Error response from daemon: No such container: chinni111/nodejs-demo-app
PS C:\Users\umama\OneDrive\Desktop\ELEVATELABS-TASK-1> docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
2d1ae4cd9c07 chinni111/nodejs-demo-app "docker-entrypoint.s..." 44 seconds ago Up 12 seconds 0.0.0.0:3000->3000/tcp elegant_franklin
PS C:\Users\umama\OneDrive\Desktop\ELEVATELABS-TASK-1> docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
chinni111/nodejs-demo-app latest 68db2990c102 24 minutes ago 134MB
registry.k8s.io/kube-apiserver v1.29.2 8a9000f98a52 17 months ago 127MB
registry.k8s.io/kube-proxy v1.29.2 934afe2372f 17 months ago 82.3MB
registry.k8s.io/kube-scheduler v1.29.2 6fc5e6b7218c 17 months ago 59.5MB
registry.k8s.io/kube-controller-manager v1.29.2 138fb5a3a2e3 17 months ago 122MB
registry.k8s.io/etcd 3.5.10-0 a0ed15eed44 21 months ago 148MB
registry.k8s.io/coredns v1.11.1 cbbe1a7bd410 23 months ago 59.8MB
docker/desktop-vpnkit-controller dc331ch22850be0cd97c84a9cfecaf44a1af6e 556998075b3d 2 years ago 36.2MB
registry.k8s.io/pause 3.9 e6f181688397 2 years ago 744KB
docker/desktop-storage-provisioner v2.0 99f89471f470 4 years ago 41.9MB

PS C:\Users\umama\OneDrive\Desktop\ELEVATELABS-TASK-1> docker stop chinni111/nodejs-demo-app
Error response from daemon: No such container: chinni111/nodejs-demo-app
PS C:\Users\umama\OneDrive\Desktop\ELEVATELABS-TASK-1> docker stop chinni111/nodejs-demo-app:latest
Error response from daemon: No such container: chinni111/nodejs-demo-app:latest
PS C:\Users\umama\OneDrive\Desktop\ELEVATELABS-TASK-1> docker stop 2d1ae4cd9c07
Error response from daemon: No such container: 2d1ae4cd9c07
PS C:\Users\umama\OneDrive\Desktop\ELEVATELABS-TASK-1> docker ps

```

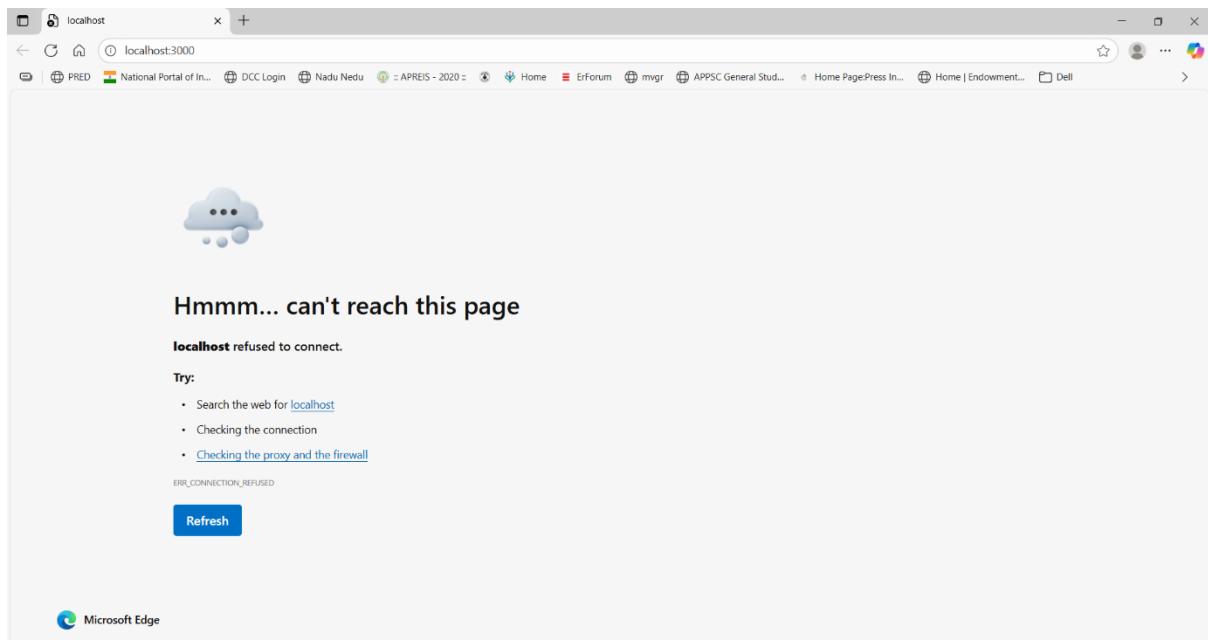


```

File Edit Selection View Go Run Terminal Help ← ↵ ELEVATELABS-TASK-1
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS ANSIBLE
EXPLORER ... REPOSITORY TAG IMAGE_ID CREATED SIZE
ELEVATELABS-TASK-1 .gitlab/workflows main.yml node_modules app.js Dockerfile package-lock.json package.json README.md
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
2d1a4cd9c07 chime111/nodejs-demo-app "docker-entrypoint.s..." 6 minutes ago Exited (1) 22 seconds ago
elegant_franklin
PS C:\Users\umma\OneDrive\Desktop\ELEVATELABS-TASK-1> docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
2d1a4cd9c07 chime111/nodejs-demo-app "docker-entrypoint.s..." 6 minutes ago Exited (1) 22 seconds ago
elegant_franklin
PS C:\Users\umma\OneDrive\Desktop\ELEVATELABS-TASK-1> docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
3.5.10-0 elegant_franklin
PS C:\Users\umma\OneDrive\Desktop\ELEVATELABS-TASK-1> docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
3.5.10-0 elegant_franklin
PS C:\Users\umma\OneDrive\Desktop\ELEVATELABS-TASK-1> docker images
REPOSITORY TAG IMAGE_ID CREATED SIZE
chime111/nodejs-demo-app latest 68db2990c102 31 minutes ago 134MB
registry.k8s.io/kube-apiserver v1.29.2 8a9000f98a52 17 months ago 127MB
registry.k8s.io/kube-proxy v1.29.2 9344fc02372f 17 months ago 82.3MB
registry.k8s.io/kube-controller-manager v1.29.2 138fb5a32e33 17 months ago 107MB
registry.k8s.io/kube-scheduler v1.29.2 6fc5e6b7218c 17 months ago 59.5MB
registry.k8s.io/etcd v1.11.1 a0ed15eed44 21 months ago 148MB
registry.k8s.io/coredns v1.11.1 cb0117bd410 23 months ago 59.8MB
docker.vpnkit-controller dc331cb22850be0cd97c84a9cfecaf44a1afbf6e 556998975b3d 2 years ago 36.2MB
registry.k8s.io/pause 3.9 e5f181688397 2 years ago 744KB
docker/desktop-app-provisioner v2.0 99fb89471f470 4 years ago 41.9MB
PS C:\Users\umma\OneDrive\Desktop\ELEVATELABS-TASK-1> docker rmi 68db2990c102
Untagged: chime111/nodejs-demo-app:latest
Untagged: chime111/nodejs-demo-app@sha256:bacd3c5e1263f57edc08737fe0f1fc93ec4be3b1c46cab8a1421429b2ad96b69
Deleted: sha256:68db2990c102db8b71689f6b56505a3018a137fe511c0da18586f9169fd70fe0de
Deleted: sha256:6731f018c6add18a76569e12acfae7fde51148e34fc40dd1aeb116dfbded2b
Deleted: sha256:643d000ef7f5376a1914961761841ab856ec5f8a70717d12a868b6b9a2e7e0157
Deleted: sha256:f8d4afdc03e528f8d4dr8e1225f12d1eb4988e6c73ce3e22f7eeff5a10f7f1a1d67
Deleted: sha256:ca0f01a750c6df3cb047c41da47e5e60820f39248f2018bb8bc5273907f267c69
Deleted: sha256:66518ab4d9173aa253206e2403d033598996dd43d8a2d4f87d874196f2c65d6fd
Deleted: sha256:360364ac7663febf5fx5c517355f5b91dcfa8a5c5b7df28677575016e4efee495
Deleted: sha256:33988a8a68ff4a123c6e91492fe1f29bd932e7a20680f489fdec4e17cfa04328
Deleted: sha256:08800c18d1fddaf9553d747a58cf4409234239ab01aa96cf263d2215b8b359
PS C:\Users\umma\OneDrive\Desktop\ELEVATELABS-TASK-1> docker images
REPOSITORY TAG IMAGE_ID CREATED SIZE
registry.k8s.io/kube-apiserver v1.29.2 8a9000f98a52 17 months ago 127MB
registry.k8s.io/kube-scheduler v1.29.2 6fc5e6b7218c 17 months ago 59.5MB
registry.k8s.io/kube-proxy v1.29.2 9344fc02372f 17 months ago 82.3MB
registry.k8s.io/kube-controller-manager v1.29.2 138fb5a32e33 17 months ago 107MB

```

After container stopped, we cannot access the application.



## Important Files

- app.js: Main Node.js server file
- Dockerfile: Instructions to build the Docker image
- main.yml: GitHub Actions workflow configuration

**CONCLUSION:** Through this task, I successfully implemented a complete CI/CD pipeline using GitHub Actions. I automated the process of testing, building, and deploying a Node.js application as a Docker image to DockerHub. Without building the image locally, I pulled it from DockerHub and ran it in a container, then accessed the running app through my local browser. This hands-on experience helped me understand practical DevOps workflows and the integration of version control, containerization, and automation.