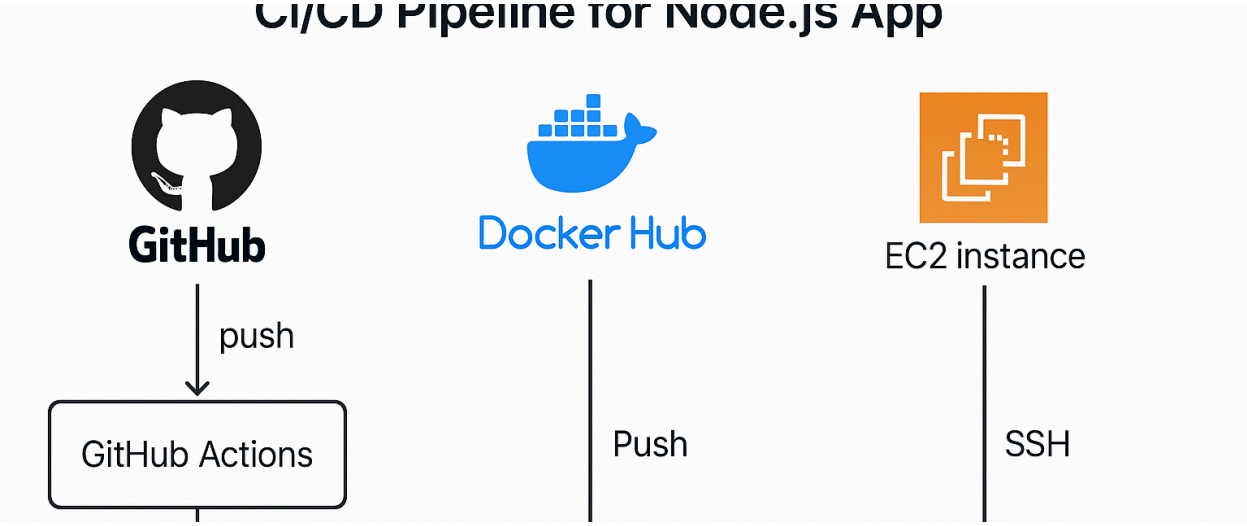


(Somasekhar)CI/CD with GitHub Actions – Deploying a Node.js App to AWS EC2 using Docker & SSH

1. Overview

In this setup I automated the deployment of a Node.js application to an AWS EC2 instance using **GitHub Actions**.

The pipeline does:

1. On every push to `main`:
2. Build the Node.js project and create a **Docker image**.
3. Push the image to **Docker Hub**.
4. SSH into the **EC2 instance**.
5. Pull the latest image and run the container.

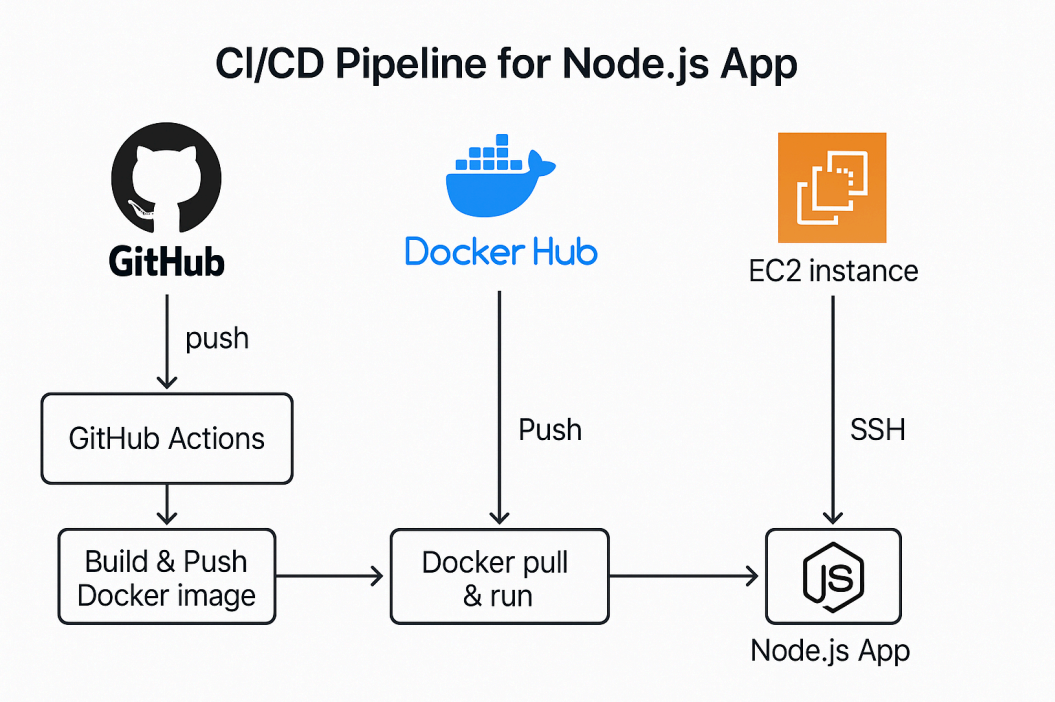
2. Architecture

- **GitHub** – Hosts application source code and GitHub Actions workflow.

- **Docker Hub** – Stores built Docker images.
- **AWS EC2 instance** – Target server where the container runs.
- **GitHub Actions** – CI/CD engine that builds, pushes, and deploys.

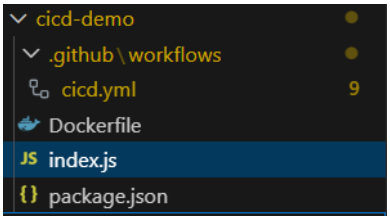
Flow:

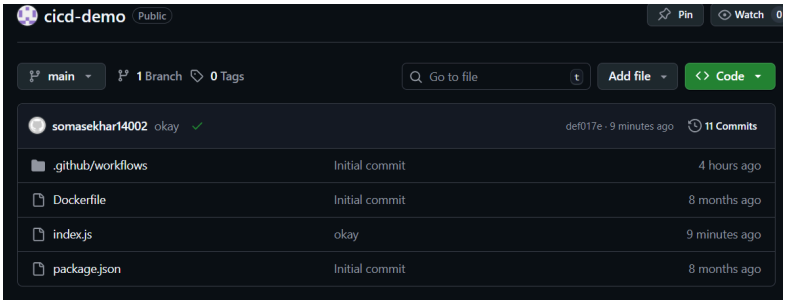
`Developer push → GitHub Actions → Build & Push Docker image → SSH to EC2 → Docker pull & run`



3. Prerequisites

1. **Node.js application** with a `package.json`.
2. **GitHub repository** with the Node.js app pushed.
3. **Docker Hub account** and a repository created (e.g. `username/node-app`).
4. **AWS EC2 instance** (Ubuntu or Amazon Linux) with:
    - Security group allowing SSH (port 22) and app port (e.g. 3000 or 80).
    - SSH key pair (.pem) downloaded.

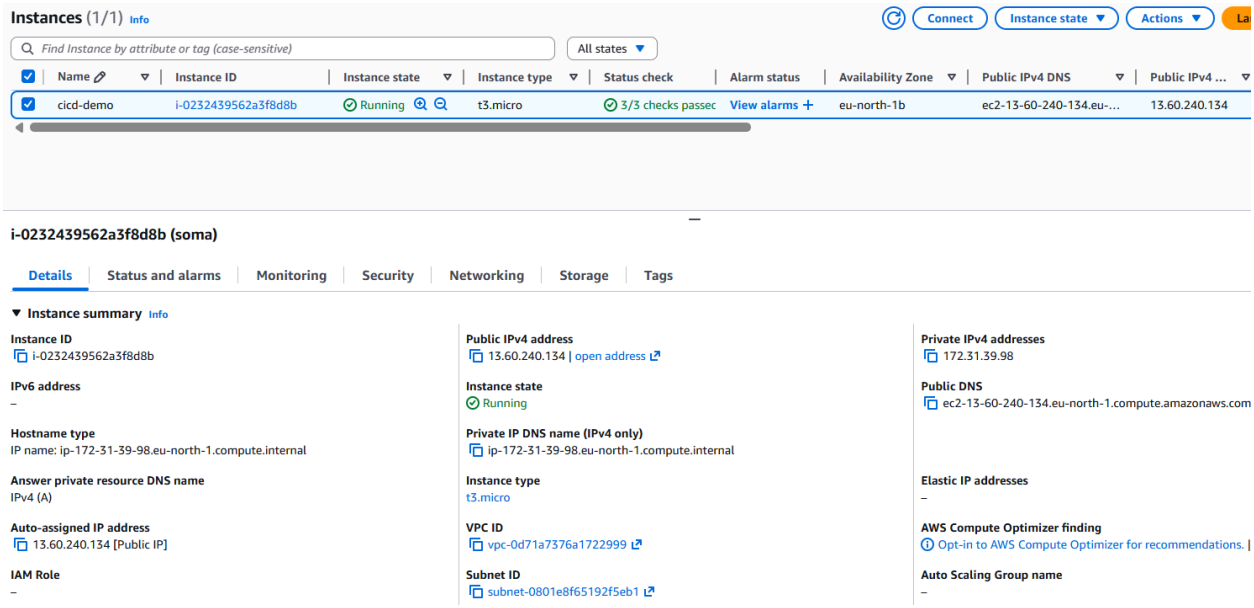




4. Prepare the EC2 instance

SSH into the instance from your local machine:

```
ssh -i your-key.pem ubuntu@<EC2_PUBLIC_IP>
```



4.1 Install Docker

```
sudo apt update
sudo apt install -y docker.io
sudo usermod -aG docker ubuntu   # or ec2-user for Amazon Linux
```

Log out & log back in so Docker group changes take effect.

(Optional) Enable Docker at start up:

```
sudo systemctl enable docker
sudo systemctl start docker
```

Now EC2 is ready to run containers.

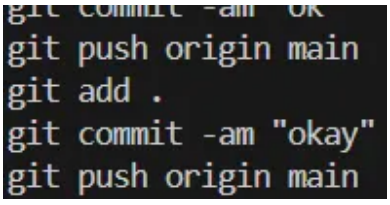
5. Containerize the Node.js application

In your project root, create a `Dockerfile`:

```
FROM node:18-alpine

WORKDIR /app

COPY package*.json ./
RUN npm install

COPY . .

EXPOSE 3000
CMD ["npm", "start"]
```

> Adjust the port and start command according to your app.

Commit and push this to GitHub.



6. Configure GitHub Secrets

Go to: **GitHub Repo → Settings → Secrets and variables → Actions → New repository secret**

Create the following secrets:

- `DOCKERHUB_USERNAME` – Your Docker Hub username
- `DOCKERHUB_TOKEN` – Docker Hub access token/password
- `EC2_HOST` – Public IP or DNS of EC2
- `EC2_USER` – `ubuntu` or `ec2-user`
- `EC2_SSH_KEY` – Content of your private key file (`.pem`)

  (Copy the full text including `-----BEGIN OPENSSH PRIVATE KEY-----` ...)

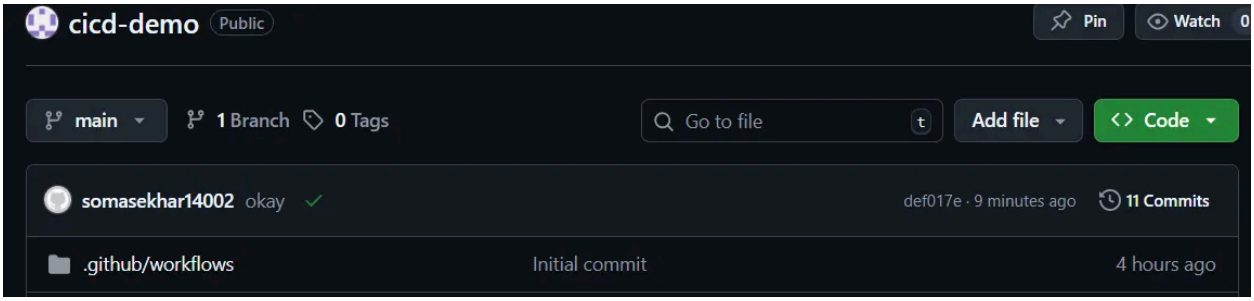
You can also add:

- `DOCKER_IMAGE` – e.g. `username/node-app`

7. Create GitHub Actions workflow

Create folder and file:

```
.github/workflows/cicd.yml
```

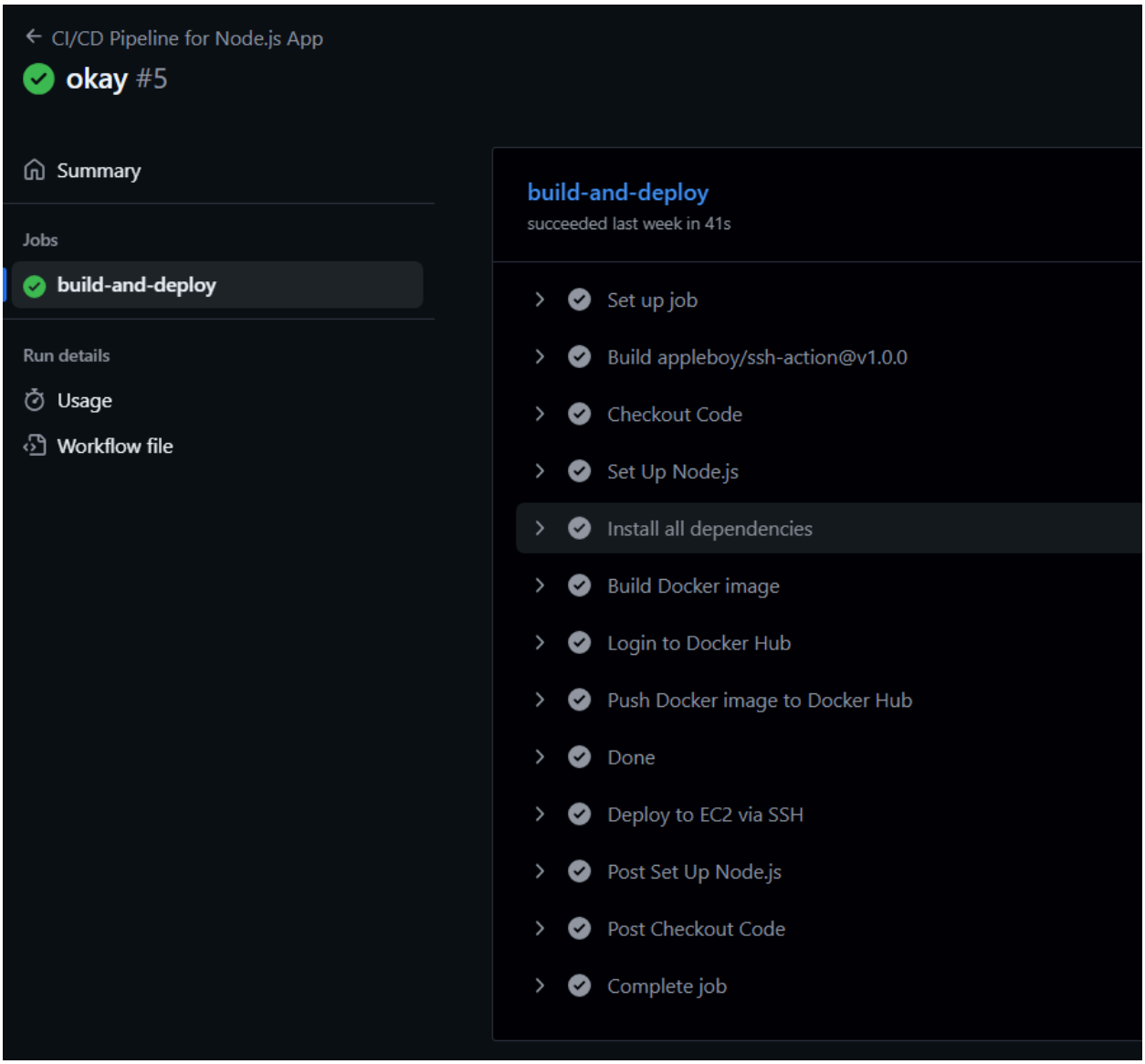


Put this YAML (adapt names as needed):

```
name: CI/CD Pipeline for Node.js App

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 all dependencies
        run: npm install

      - name: Build Docker image
        run: docker build -t ${{ secrets.DOCKER_USERNAME }}/my-node-ap
p:latest .
```

```
      - name: Login to Docker Hub
        run: echo "${{ secrets.DOCKER_PASSWORD }}" | docker login -u "${{
secrets.DOCKER_USERNAME }}" --password-stdin

      - name: Push Docker image to Docker Hub
        run: docker push ${{ secrets.DOCKER_USERNAME }}/my-node-app:la
test

      - name : Done
        run: echo "Docker Image to Docker Hub!"

      - name: Deploy to EC2 via SSH
        uses: appleboy/ssh-action@v1.0.0
        with:
          host: ${{ secrets.EC2_HOST }}
          username: ${{ secrets.EC2_USER }}
          key: ${{ secrets.EC2_SSH_KEY }}
          script: |
            docker pull ${{ secrets.DOCKER_USERNAME }}/my-node-app:late
st
            docker stop myapp || true && docker rm myapp || true
            docker run -d --name myapp -p 3000:3000 ${{ secrets.DOCKER_
USERNAME }}/my-node-app:latest
```

This matches the stages you see in below screenshot:



GitHub automatically adds “Build appleboy/ssh-action@v1.0.0”, “Post Checkout Code”, “Post Set Up Node.js”, and “Complete job” around these steps.

8. How the deployment step works (SSH to EC2)

In the `Deploy to EC2 via SSH` step, `appleboy/ssh-action`:

1. Uses the EC2 host, username, and private key from **GitHub Secrets**.
2. Opens an SSH session to the EC2 instance from the GitHub Actions runner.
3. Runs the `script` block on the EC2 server:

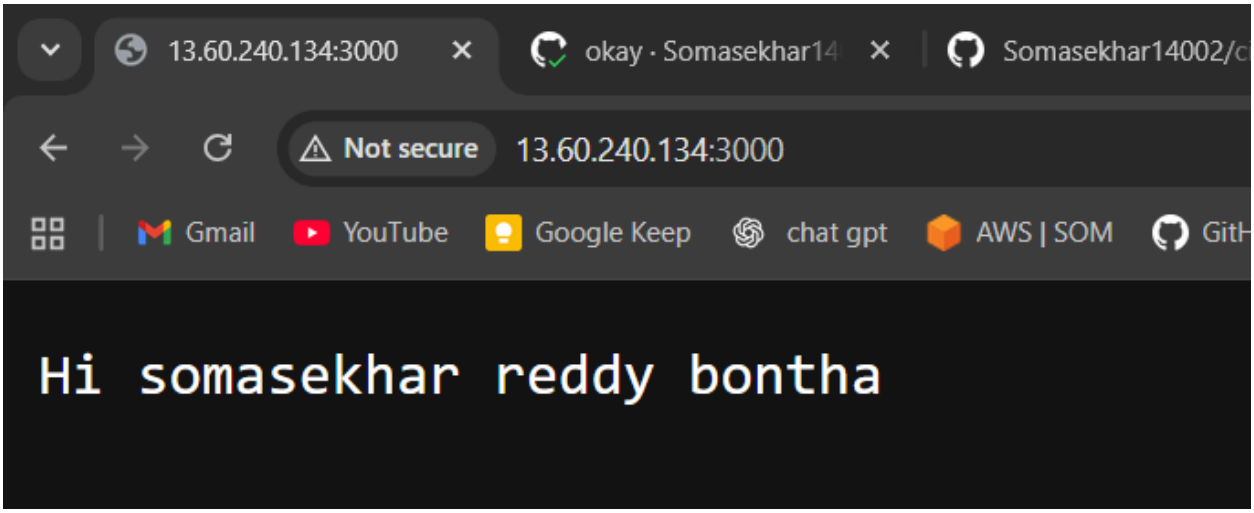
- `docker pull` – gets the latest image that was just pushed.
- Stops/removes any existing container named `node-app`.
- Runs a new container with the new image and maps container port `3000` to host port `3000`.

Result: the application on EC2 is updated automatically whenever you push to `main`.

---

9. Testing the pipeline

1. Commit and push a change to the `main` branch.
2. Go to **GitHub → Actions → CI/CD Pipeline for Node.js App**.
3. Open the latest run. You should see a job similar to:
   - Set up job
   - Checkout Code
   - Set Up Node.js
   - Install all dependencies
   - Build Docker image
   - Login to Docker Hub
   - Push Docker image to Docker Hub
   - Deploy to EC2 via SSH
   - Complete job ✅
4. After it succeeds, hit `http://<EC2_PUBLIC_IP>:3000` in the browser and verify the app.


13.60.240.134:3000
okay · Somasekhar14
Somasekhar14002/c
Not secure
13.60.240.134:3000
Gmail
YouTube
Google Keep
chat gpt
AWS | SOM
GitH
Hi somasekhar reddy bontha