**Devops Task -1**

**Step-by-Step Guide: Automate Node.js App Deployment with GitHub Actions + DockerHub (Fully Online)**

**Step 0: Create Accounts (if you don’t have them)**

1. **GitHub Account:**
   * Go to [github.com](https://github.com/)
   * Sign up for a free account.
2. **DockerHub Account:**
   * Go to [hub.docker.com](https://hub.docker.com/)
   * Sign up for a free account.

**Step 1: Create Your Node.js App Repository on GitHub**

1. Login to GitHub.
2. Click the **+** icon (top right) → **New repository**.
3. Fill in:
   * Repository name: nodejs-demo-app
   * Description: (optional) Demo Node.js app for CI/CD
   * Public or Private: your choice
   * Initialize with a README: yes (optional)
4. Click **Create repository**.

**Step 2: Add Your Node.js App Code (Using GitHub Web Editor)**

You’ll need two files at minimum:

**2.1 Create package.json**

* Go to your repo page.
* Click **Add file → Create new file**.
* Name the file package.json.
* Paste this example content:

{

"name": "nodejs-demo-app",

"version": "1.0.0",

"scripts": {

"start": "node index.js",

"test": "echo \"Running tests...\" && exit 0"

},

"dependencies": {}

}

* Scroll down and commit the new file.

**2.2 Create index.js**

* Click **Add file → Create new file** again.
* Name it index.js.
* Paste this simple code:

const http = require('http');

const port = 3000;

const requestHandler = (req, res) => {

res.end('Hello from Node.js app!');

};

const server = http.createServer(requestHandler);

server.listen(port, () => {

console.log(`Server running on port ${port}`);

});

* Commit the file.

**Step 3: Add a Dockerfile**

1. Click **Add file → Create new file**.
2. Name it Dockerfile (no extension).
3. Paste this content:

FROM node:16

WORKDIR /app

COPY package\*.json ./

RUN npm install

COPY . .

EXPOSE 3000

CMD ["npm", "start"]

1. Commit the file.

**Step 4: Add the GitHub Actions Workflow**

1. In your repo, click **Add file → Create new file**.
2. Name the file: .github/workflows/main.yml

(Note: You must type the entire path including folders. GitHub will auto-create folders.)

1. Paste this workflow YAML:

name: CI/CD Pipeline

on:

push:

branches:

- main

jobs:

build-and-deploy:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v3

- name: Setup Node.js

uses: actions/setup-node@v3

with:

node-version: '16'

- run: npm install

- run: npm test

- name: Log in to DockerHub

uses: docker/login-action@v2

with:

username: ${{ secrets.DOCKERHUB\_USERNAME }}

password: ${{ secrets.DOCKERHUB\_TOKEN }}

- run: docker build -t ${{ secrets.DOCKERHUB\_USERNAME }}/nodejs-demo-app:latest .

- run: docker push ${{ secrets.DOCKERHUB\_USERNAME }}/nodejs-demo-app:latest

1. Commit the file.

**Step 5: Add DockerHub Credentials as GitHub Secrets**

1. Log in to DockerHub:
   * Go to [hub.docker.com](https://hub.docker.com/)
   * Log in.
2. Create an access token (recommended over password):
   * Click your profile icon → **Account Settings** → **Security** → **New Access Token**.
   * Name it something like GitHub Actions Token.
   * Generate and copy the token.

(Run

docker login -u sneha2030

2. At the password prompt, enter the personal access token.

dckr\_pat\_yRDWsQ8oLfAJpggMsuARgL4V6qo)

1. Go back to your GitHub repository.
2. Click **Settings** tab → **Secrets and variables** → **Actions**.
3. Click **New repository secret**.
4. Create these two secrets:
   * Name: DOCKERHUB\_USERNAME  
     Value: your DockerHub username
   * Name: DOCKERHUB\_TOKEN  
     Value: the DockerHub password OR the access token you just created

**Step 6: Trigger the Pipeline**

1. Make sure your default branch is named main (GitHub default is usually main).
2. Make any commit or just push the files you created if not already committed.
3. The workflow will automatically trigger on push to main.

**Step 7: Monitor Workflow Run**

1. In your GitHub repo, click on the **Actions** tab.
2. You will see your workflow running.
3. Click on the latest workflow to see the logs for each step (checkout, setup node, install, test, docker login, build, push).

A screenshot of a computer

AI-generated content may be incorrect.

**Step 8: Verify Docker Image on DockerHub**

1. Go to DockerHub.
2. Search for your repository: yourusername/nodejs-demo-app.
3. You should see the latest pushed image.

A screenshot of a computer

AI-generated content may be incorrect.