

ELEVATE LABS INTERNSHIP TASK DAY - 1

Task Checklist Review

Task Requirement	Status	Notes
Set up CI/CD pipeline with GitHub Actions	 Completed	created and tested .github/workflows/main.yml.
Use Node.js app as base	 Completed	used a sample Node.js app (nodejs-demo-app).
Automate: test → build → push	 Completed	GitHub Actions does all three in order.
Configure DockerHub for image deployment	 Completed	Docker image successfully pushed to Docker Hub via GitHub Actions.
Trigger on push to main	 Completed	Confirmed from GitHub Actions trigger.
Deliverable: GitHub repo with CI/CD workflow	 Completed	You now have a working repo + YAML pipeline.

Deliverables

-  .github/workflows/main.yml file created and working
-  Node.js project set up
-  Docker image automatically built and pushed
-  Pipeline triggered on code push
-  Deployment automation verified via GitHub Actions UI

CREATED EC2 INSTANCE WITH UBUNTU AMI AND CONNECTED IT WITH MOBAXTERM

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with navigation links like Dashboard, EC2 Global View, Events, Instances (with sub-links for Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations), Images (AMIs, AMI Catalog), and Elastic Block Store (Volumes). The main area displays 'Instances (1/1) Info' for a single instance named 'Task-1'. The instance details are as follows:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
Task-1	i-06a707554493433b4	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	ec2-3-84-22-139.compute-1.amazonaws.com

Below the instance details, there's a tabbed interface with 'Details' selected, followed by Status and alarms, Monitoring, Security, Networking, Storage, and Tags.

The screenshot shows a MobaXterm window titled '3.84.22.139 (ubuntu)'. The terminal session is connected to the EC2 instance. The session list on the left shows a single entry for 'ubuntu@3.84.22.139 (ubuntu)'. The main pane displays the Ubuntu 24.04.2 LTS welcome screen and system information. At the bottom, a command-line session is shown:

```
Last login: Mon Apr 7 10:16:06 2025 from 106.196.28.115
ubuntu@ip-172-31-86-102:~$ ls
nodejs-s-demo-app
ubuntu@ip-172-31-86-102:~/nodejs-s-demo-app$ cd nodejs-s-demo-app
ubuntu@ip-172-31-86-102:~/nodejs-s-demo-app$ ls
Dockerfile index.js node_modules package-lock.json package.json
ubuntu@ip-172-31-86-102:~/nodejs-s-demo-app$
```

ALLOW THE TRAFFIC AND SET THE PORT RANGE

The screenshot shows the AWS EC2 Security Groups interface for editing inbound rules. The top navigation bar includes the AWS logo, search bar, and account information (United States (N. Virginia) and Soneeya SS). The main content area is titled "Edit inbound rules" with a "Info" link. A sub-header states: "Inbound rules control the incoming traffic that's allowed to reach the instance." Below this, a table lists three existing rules:

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-0bd7fc4f40f6bcd9b	All traffic	All	All	Custom	sg-0b50a0704cf66a803 X
sgr-0dc83420c27f13343	Custom TCP	TCP	3000	Custom	0.0.0.0/0 X
sgr-04b54ae8181995641	SSH	TCP	22	Custom	0.0.0.0/0 X

At the bottom left is a blue "Add rule" button. A yellow warning banner at the bottom states: "⚠️ Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only." At the very bottom are links for CloudShell, Feedback, Copyright (© 2025, Amazon Web Services, Inc. or its affiliates), Privacy, Terms, and Cookie preferences.

GITHUB REPO WITH CI/CD WORKFLOW

The screenshot shows the GitHub Actions CI/CD pipeline interface for the repository "soneeya-it21 / nodejs-demo-app". The top navigation bar includes the repository name, search bar, and various actions like Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The "Actions" tab is selected. On the left, a sidebar shows sections for Actions, CI/CD Pipeline, Management, Caches, Attestations, Runners, Usage metrics, and Performance metrics. The main content area is titled "All workflows" and shows "1 workflow run". The run details are as follows:

Event	Status	Branch	Actor
main	1 hour ago	main	39s

The run was triggered by "Add GitHub Actions CI/CD pipeline" and completed successfully. The commit pushed was "Commit 86b51ff pushed by soneeya-it21".

The screenshot shows the GitHub Actions CI/CD pipeline interface. At the top, there are navigation links: Code, Issues, Pull requests, Actions (which is highlighted), Projects, Wiki, Security, Insights, and Settings. Below this, a header bar includes a back arrow, the text "CI/CD Pipeline", a green checkmark icon, and the text "Add GitHub Actions CI/CD pipeline #1". On the right of the header are buttons for "Re-run all jobs", "Latest #4", and three dots. The main area is titled "build-and-deploy" and shows a success message "succeeded 1 hour ago in 28s". A search bar at the top right says "Search logs". The pipeline details section lists the following steps:

- > Set up job
- > Checkout source code
- > Set up Node.js
- > Install dependencies
- > Run tests
- > Log in to DockerHub
- > Build Docker image
- > Push Docker image
- > Post Set up Node.js
- > Post Checkout source code
- > Complete job

Each step has a timestamp next to it: 0s, 1s, 1s, 2s, 0s, 0s, 19s, 2s, 0s, 0s, 0s.

DOCKERHUB IMAGE DEPLOYMENT

The screenshot shows the Docker Hub repository page for the user "soneeya". The repository name is "nodejs-demo-app". The page includes a sidebar with options like Repositories, Settings, Default privacy, Notifications, Billing, Usage, Pulls, and Storage. The main content area shows the repository details:

- General** tab is selected.
- Tags** section: "This repository contains 1 tag(s)." A table shows one tag:

Tag	OS	Type	Pulled	Pushed
latest	Node.js	Image	less than 1 day	about 1 hour
- Docker commands**: "To push a new tag to this repository: docker push soneeya/nodejs-demo-app:tagname".
- Build with Docker Build Cloud**: "Accelerate image build times with access to cloud-based builders and shared cache." It also mentions "Docker Build Cloud executes builds on optimally-dimensioned cloud infrastructure with dedicated per-organization isolation." and "Get faster builds through shared caching across your team, native multi-platform support, and encrypted data transfer - all without managing infrastructure."

NODE.JS APP BASE



Hello from Node.js!