Task 2: Jenkins CI/CD Pipeline with Docker

Objective:

To create a CI/CD pipeline using Jenkins to build, test, and deploy a simple web application containerized using Docker, and run it on an AWS EC2 instance.

Tools Used

- AWS EC2 Ubuntu Instance
- Jenkins
- Docker
- GitHub
- Node.js (for the app)
- Jenkins Plugins: Git, Docker, NodeJS, Pipeline, Pipeline Stage View

Definitions

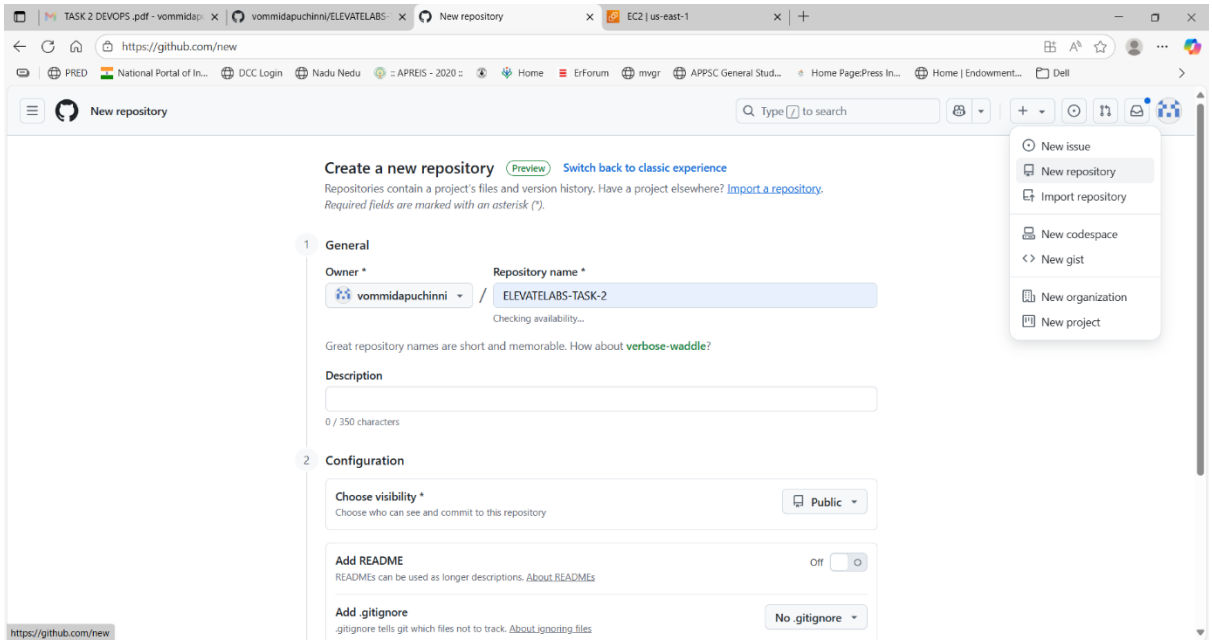
- **CI/CD**: Continuous Integration and Continuous Deployment – a process where code changes are automatically built, tested, and deployed.
- **Jenkins**: An open-source automation server used to implement CI/CD pipelines.
- **Docker**: A tool that allows applications to run in isolated containers.
- **EC2**: AWS virtual server where Jenkins and the application are hosted.
- **GitHub**: A version control platform where the code is stored.
- **Webhook** (optional): It automatically notifies Jenkins of code changes in GitHub to trigger the pipeline.

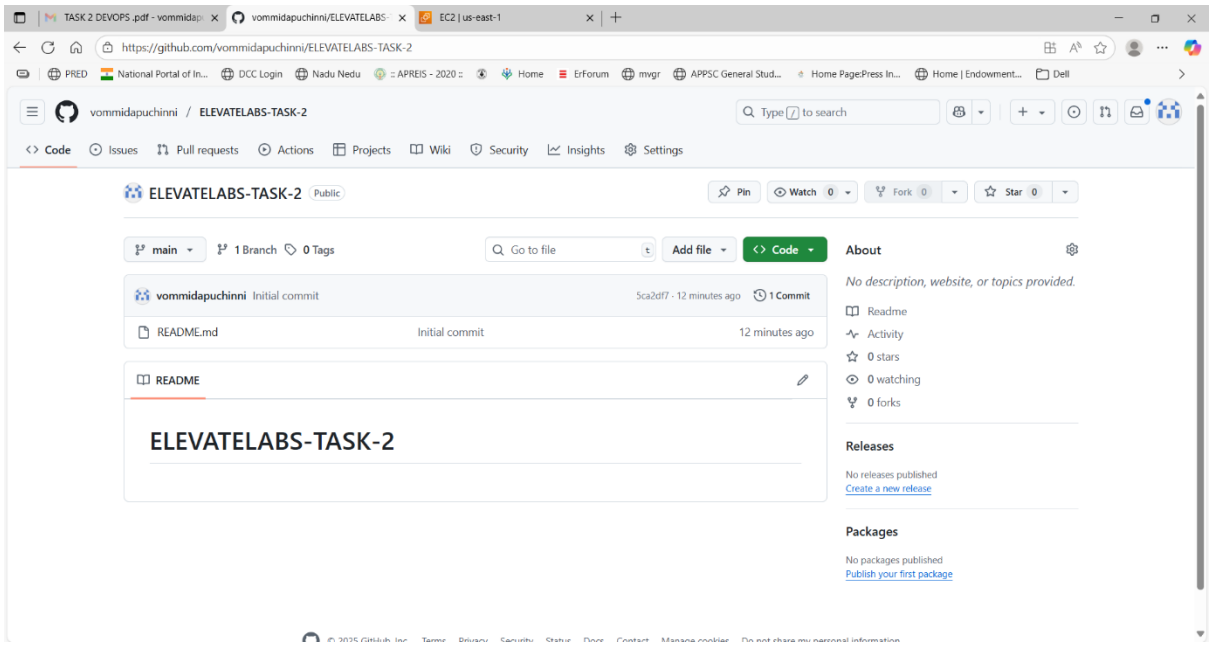
What I Did

1. Created a GitHub Repository

Click on + new repository.

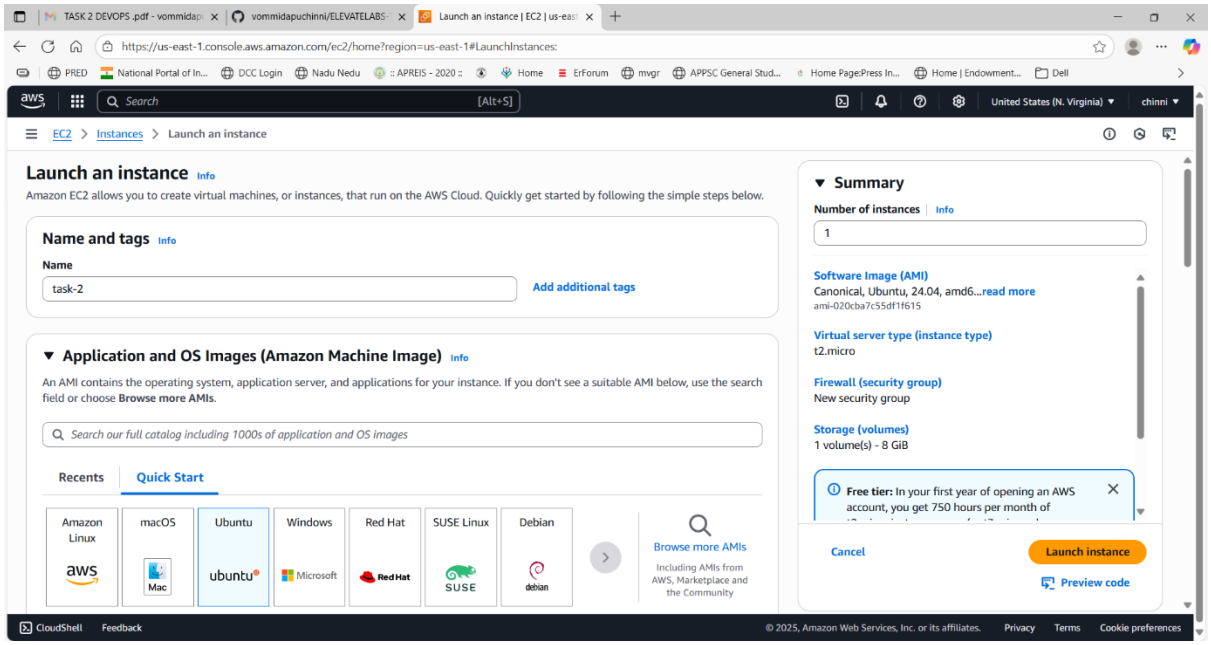
Made a public repository and added a README file to it.



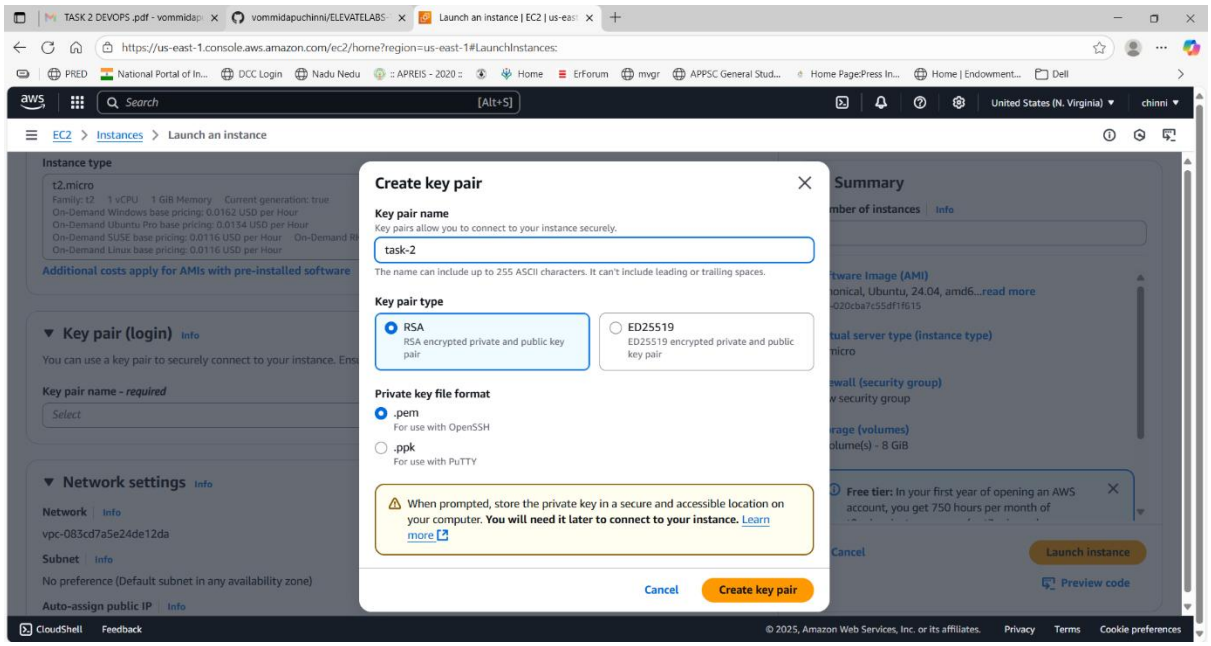


2. Launched EC2 Instance

- Login to AWS console management
- Click on launch instance
- Choose AMI as ubuntu (Linux distribution)
- Click on create new key pair
- Give name .pem is normal key .py is putty key pair.

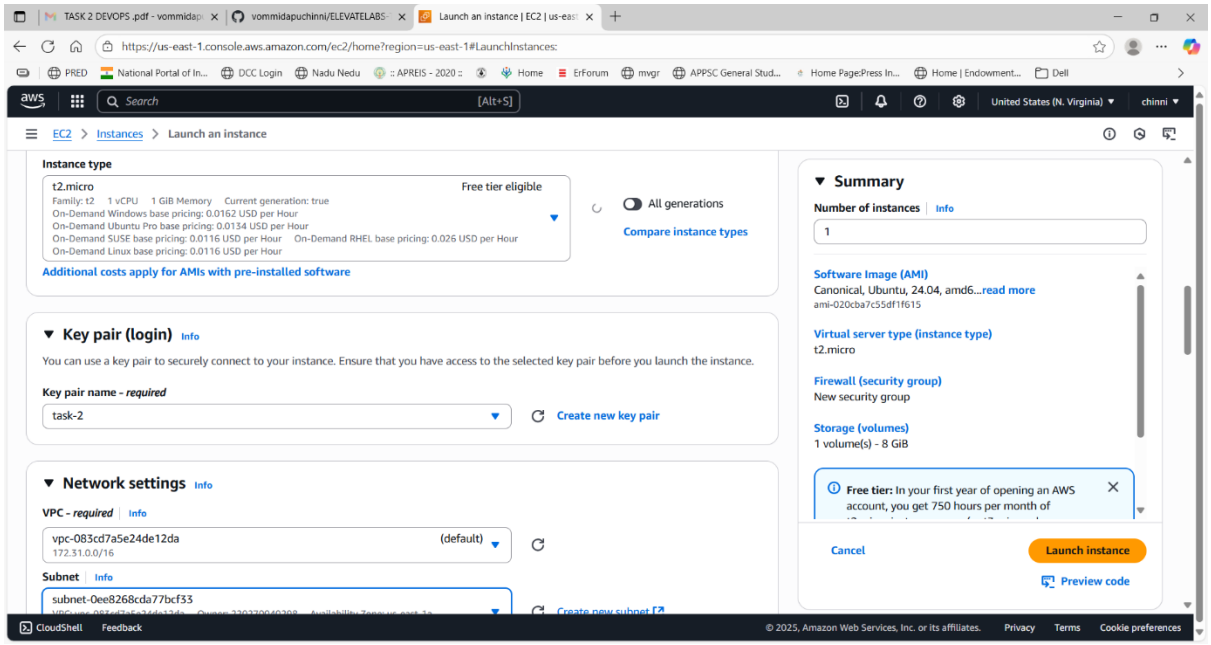


- Ubuntu 22.04 t2.micro instance.

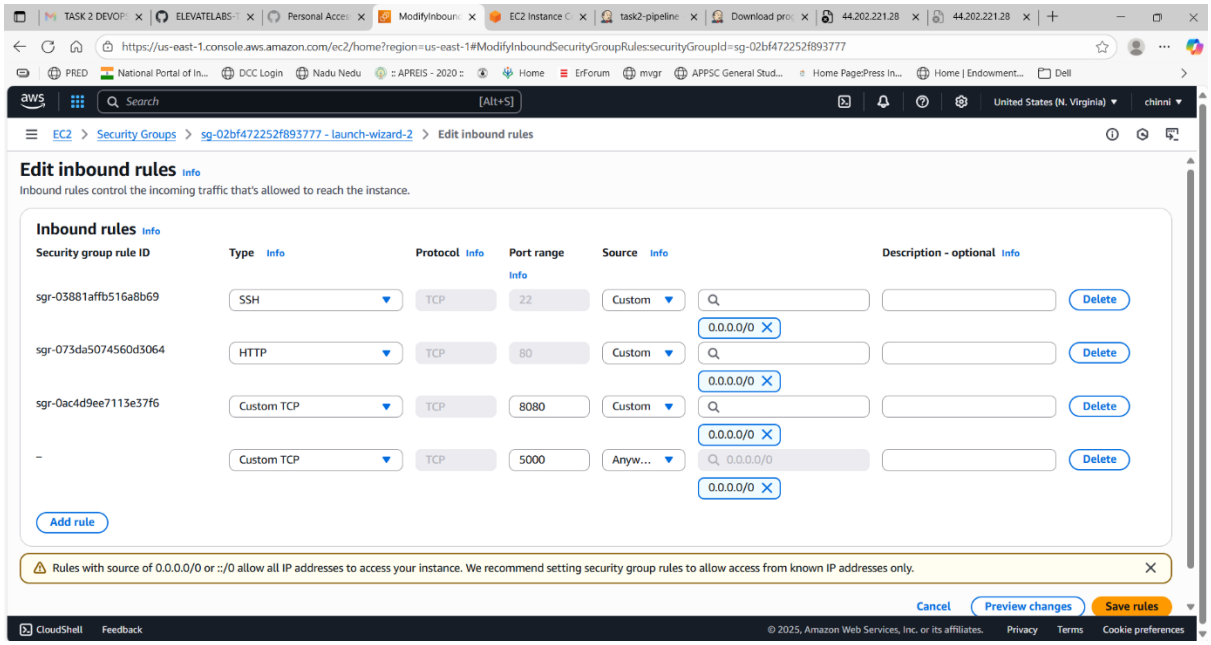


Choose VPC as default one.

Choose any subnet out of subnets.



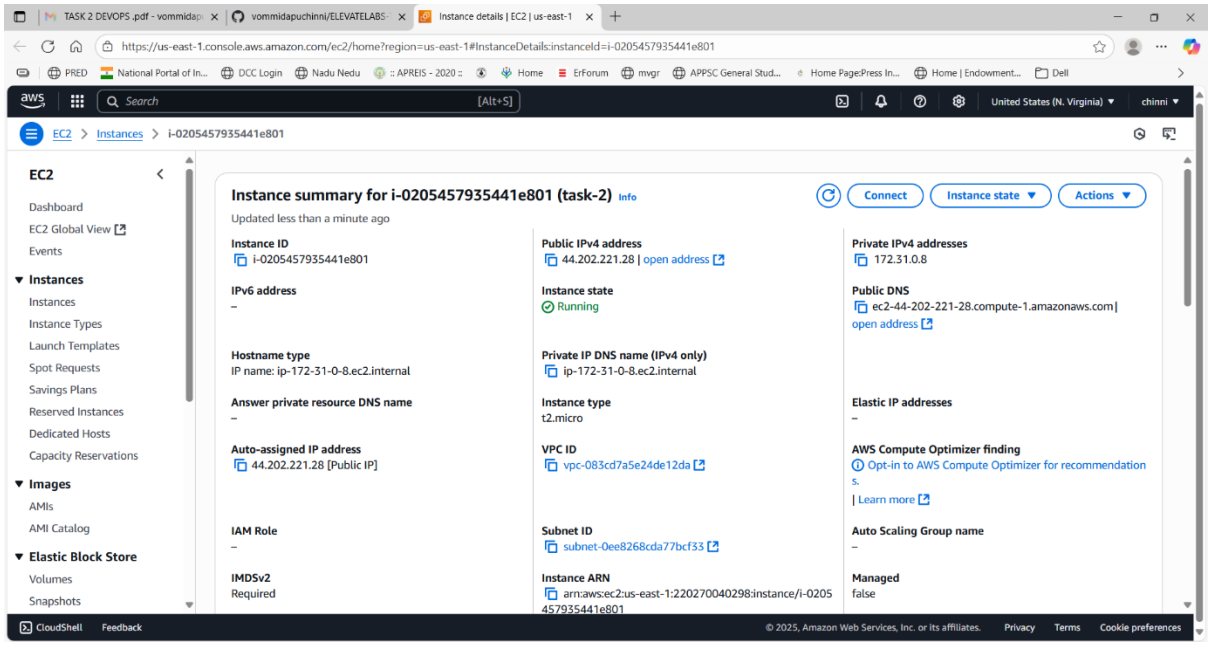
- Allowed port **5000, 8080, 80** in the security group.

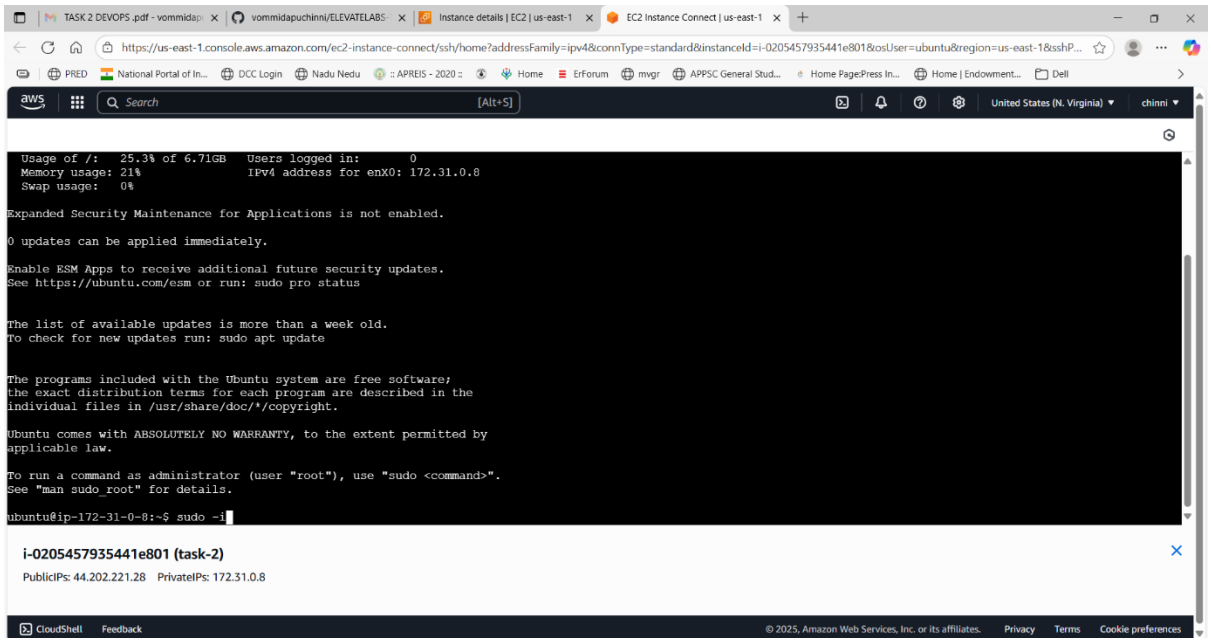


After launching an instance, we can see like this.

Click on connect to connect to an instance.

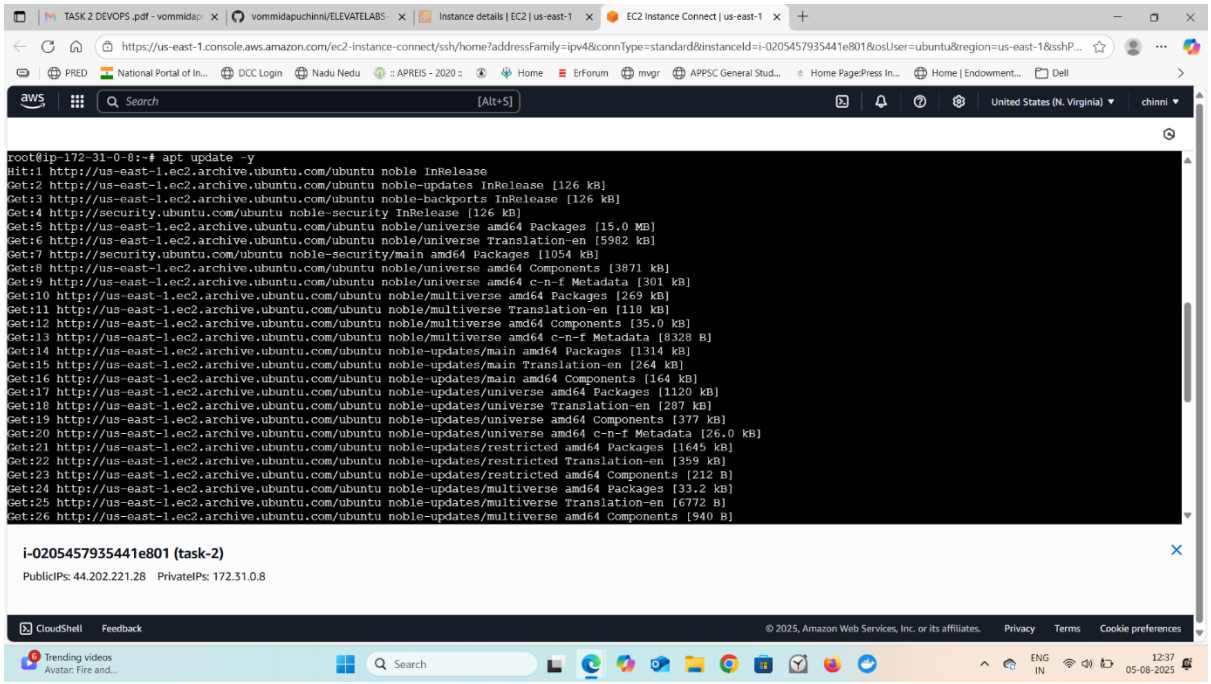
After again click on connect.



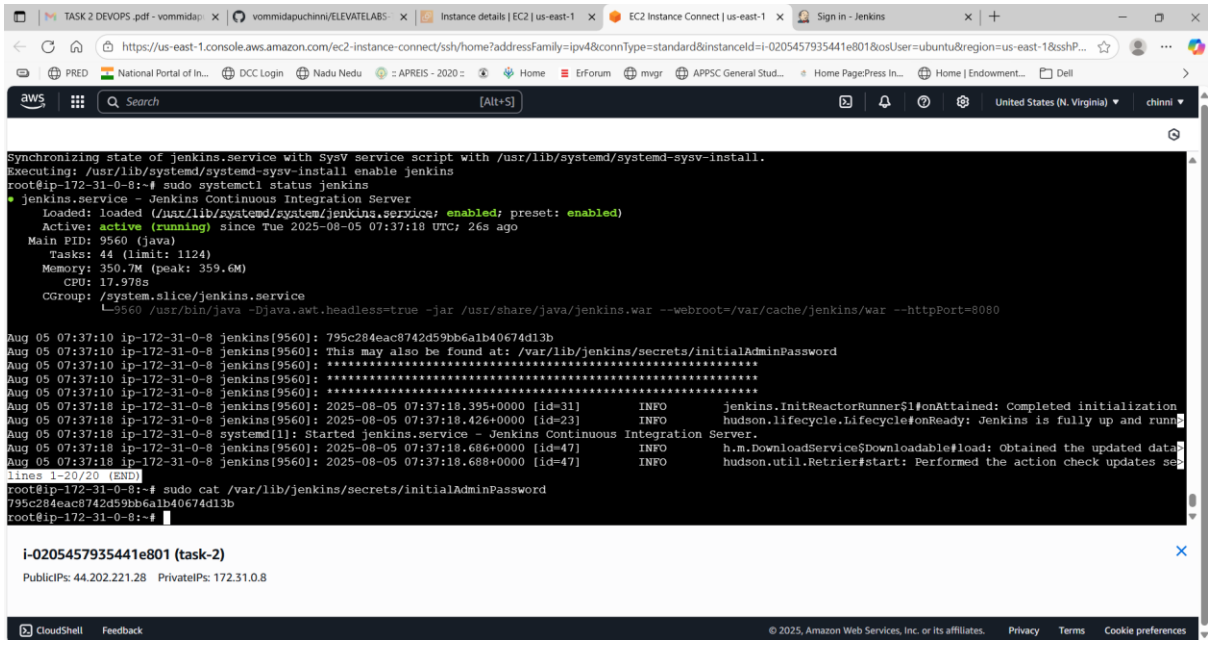


sudo -i is used to change normal user to root user.

apt update -y is used to update

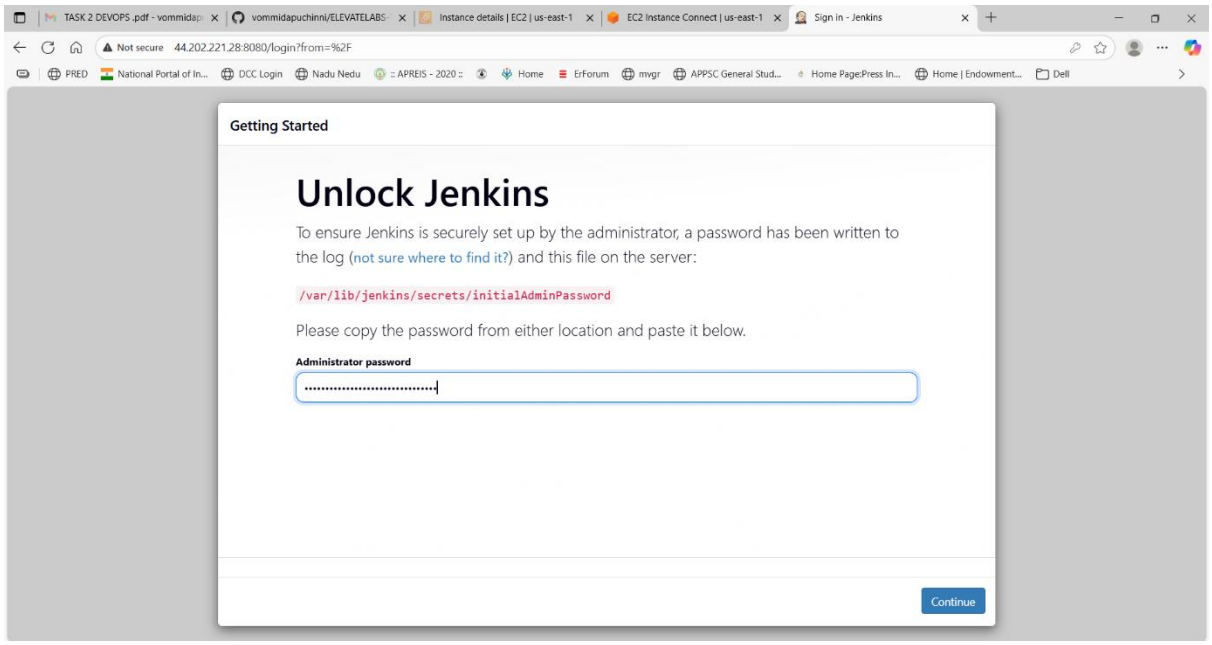


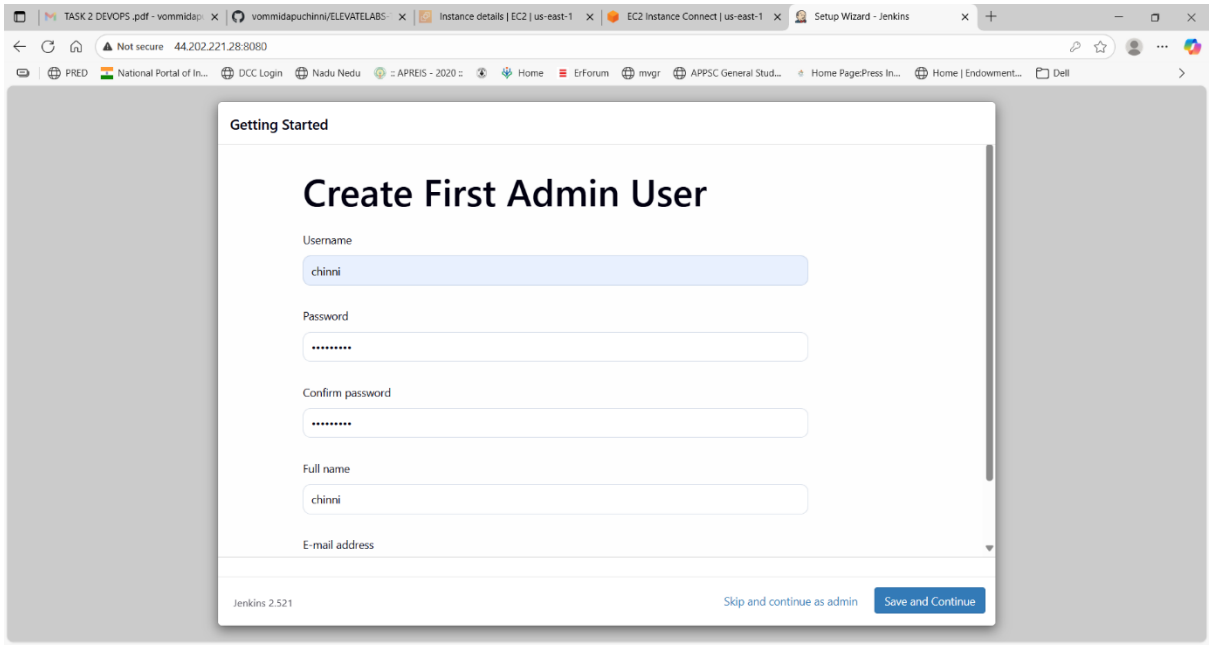
- Installed required tools:
  - Jenkins
  - Docker
  - Git
  - Node.js

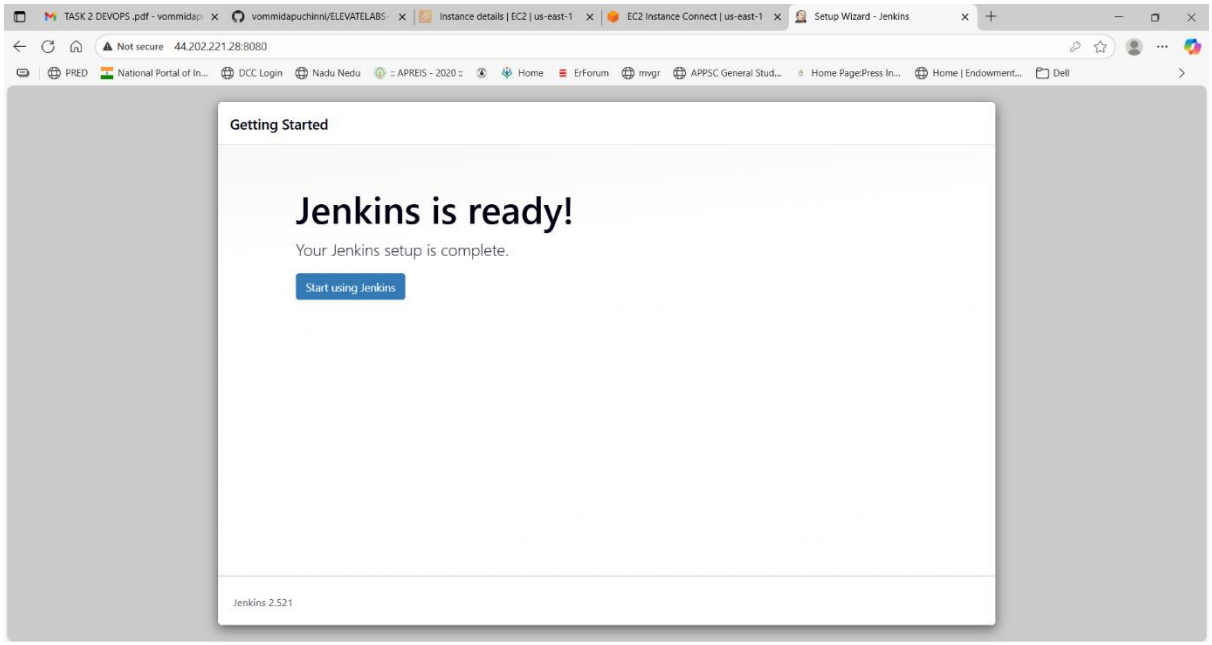


- Updated the system and installed Jenkins using commands

- Installed Java (required for Jenkins)
- Started Jenkins and accessed it from browser using http://<EC2_PUBLIC_IP>:8080
- **Note**: Used sudo cat /var/lib/jenkins/secrets/initialAdminPassword to get initial password
- systemctl status Jenkins →used to see the status of Jenkins.
- Added jenkins user to the docker group to allow Jenkins to run Docker commands

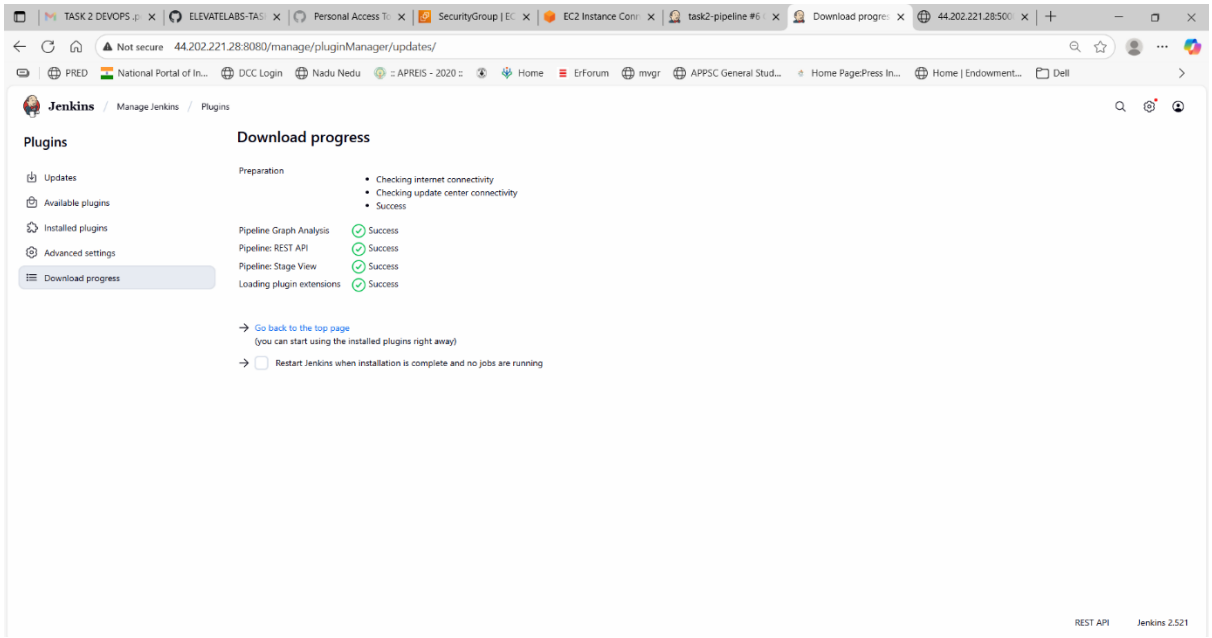






3. Install Jenkins Plugins

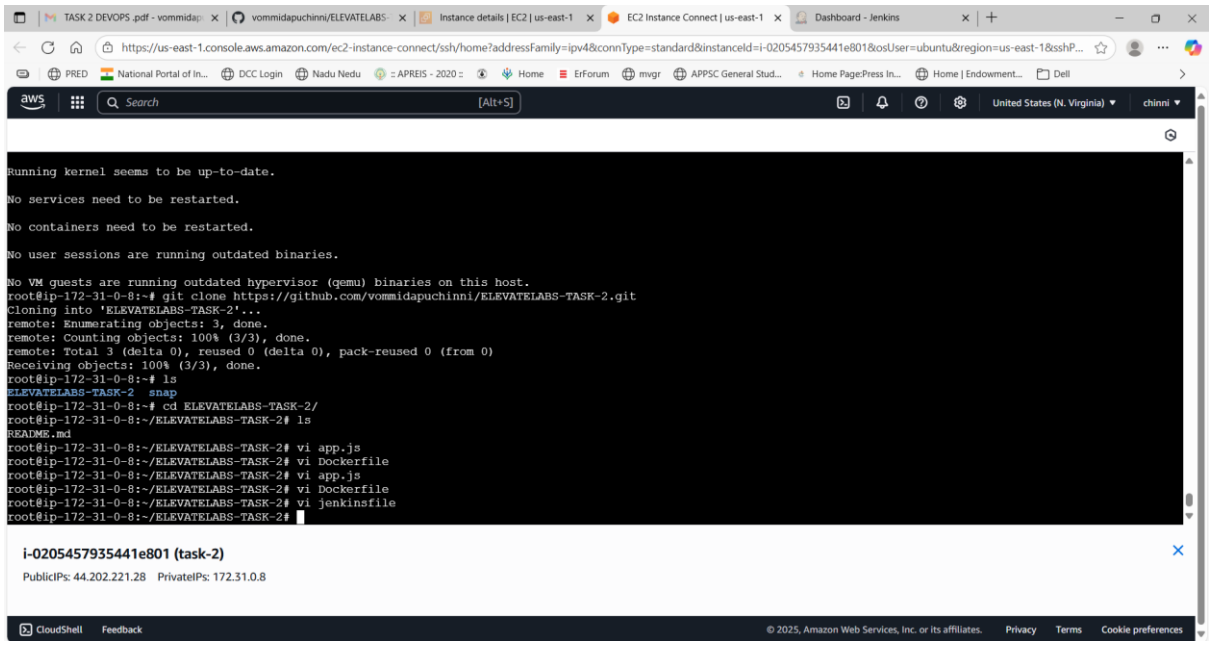
- Installed the following plugins:
  - Git
  - Docker Pipeline
  - Pipeline
  - GitHub Integration



Jenkins plugins add new features and integrations to Jenkins for different tools like Git, Docker, etc.

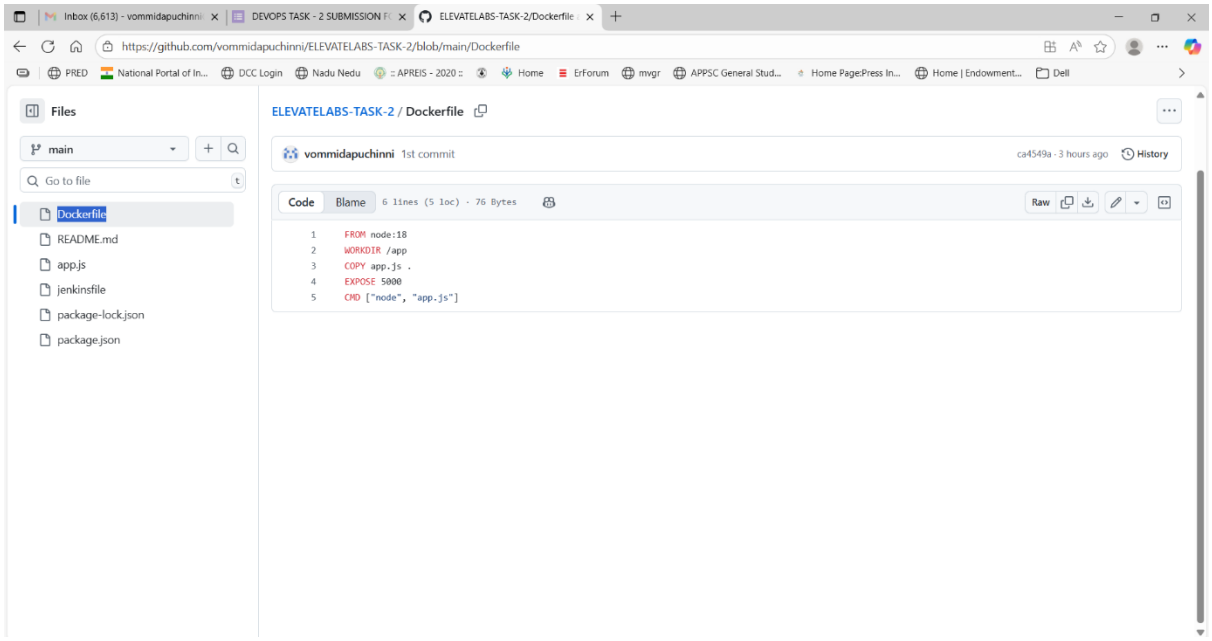
4. app.js

- A simple app with one endpoint:
- It contains an Express server with a single route
- Listens on port 5000
- **Purpose:** To test and deploy a working Node.js app via Jenkins CI/CD.
- GET / returns: Node.js App Deployed via Jenkins!



5. Dockerfile: This file is used to build a Docker image for your Node.js app

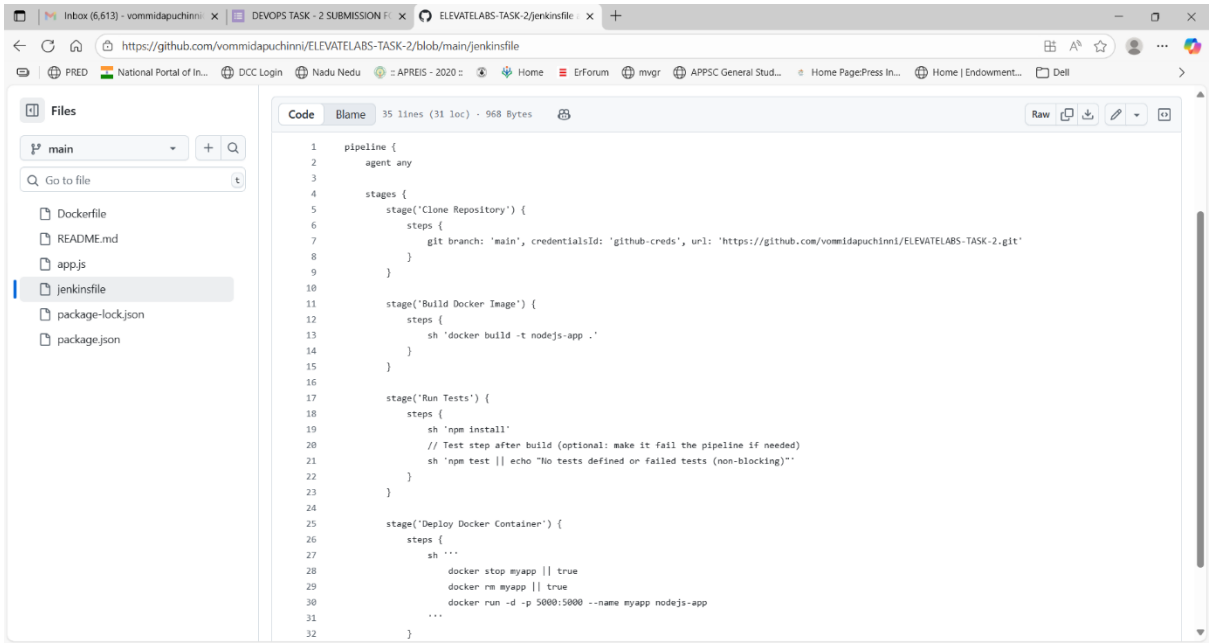
- Uses Node 18 base image
- Copies code
- Exposes port 5000
- Starts the app



6. Jenkinsfile (CI/CD Pipeline): Defines your CI/CD pipeline using stages:

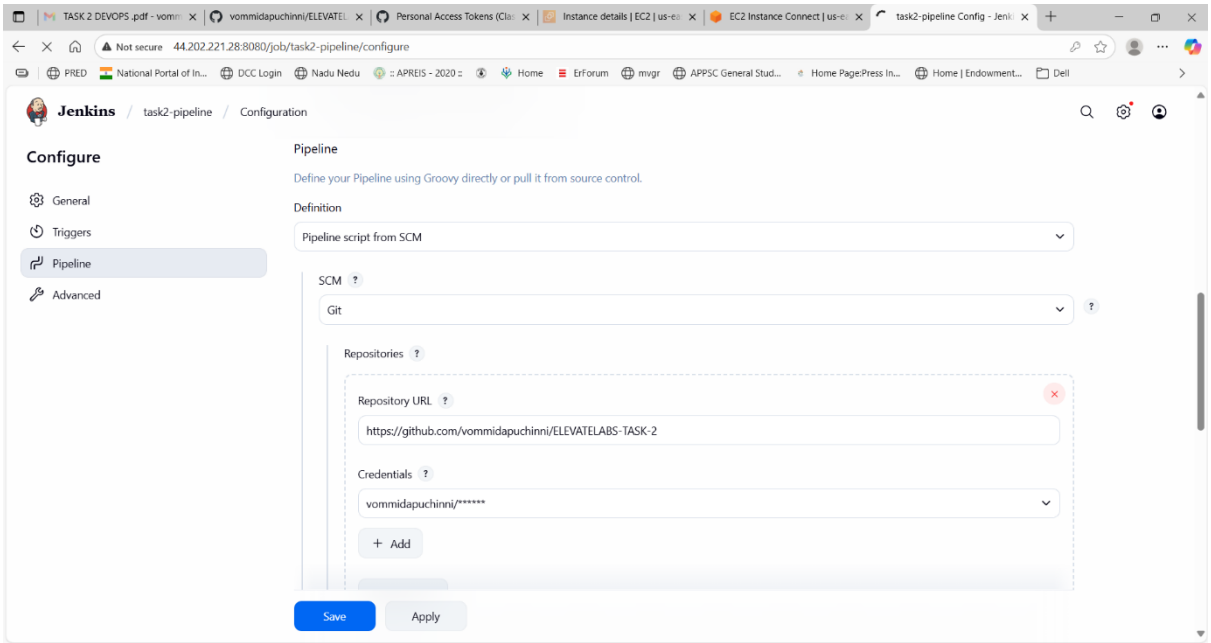
Stages:

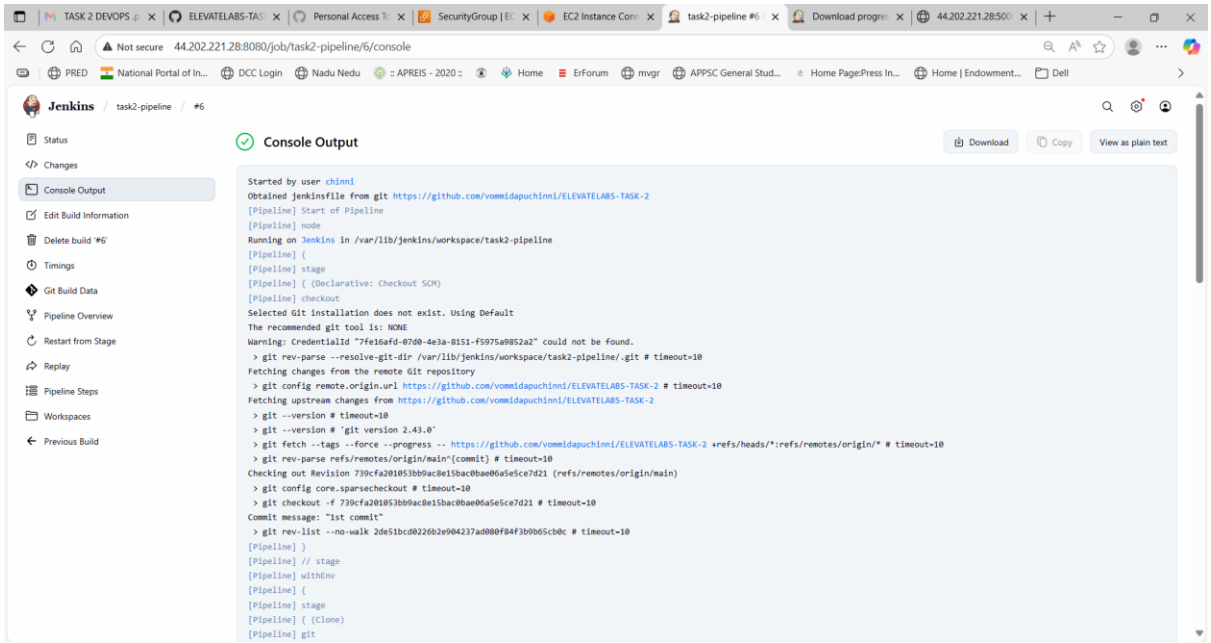
- Clone: Pull code from GitHub
- Build: Build Docker image
- Install & Test: Install deps and test
- Deploy: Run container on port 5000



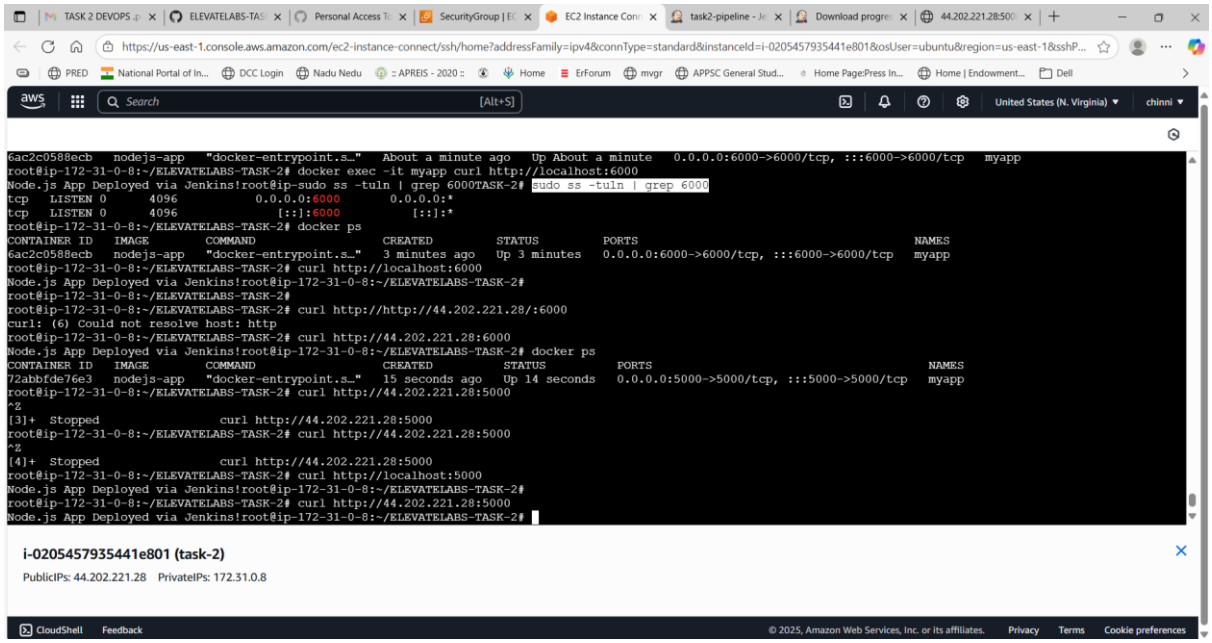
7. Configure Jenkins Pipeline Job

- Created a new Jenkins pipeline project
- Selected "Pipeline script from SCM"
- Set SCM to Git and provided GitHub repo URL
- Chose branch: main

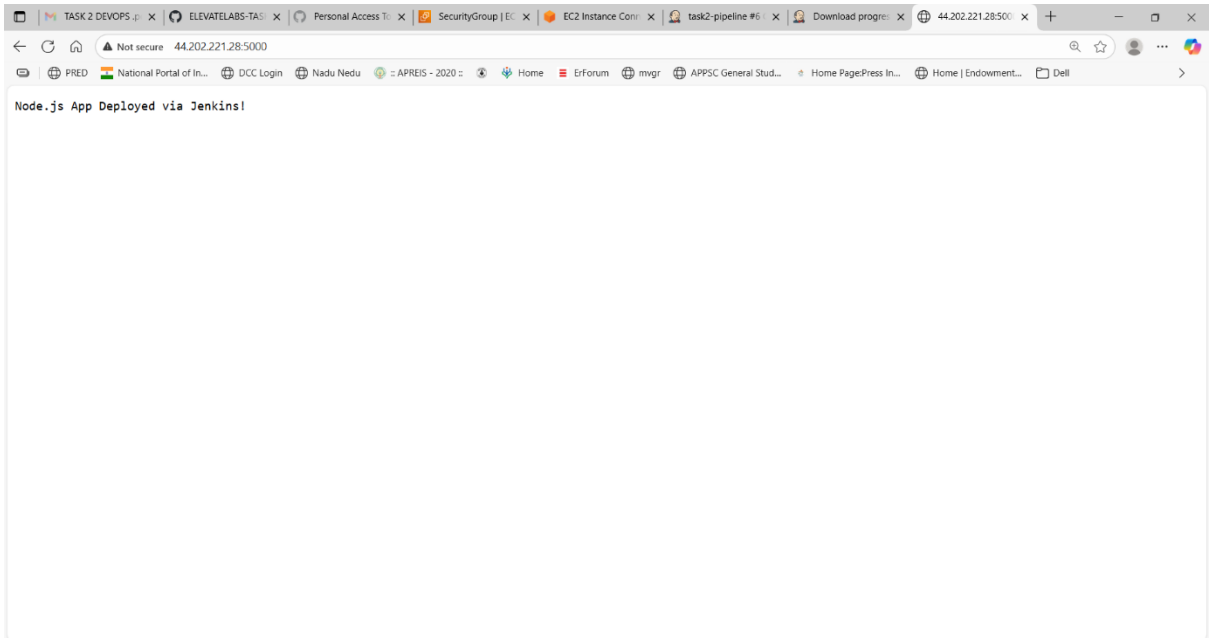




8. Access the App



- Browser: http://<ec2-ip>:5000
- Confirmed the app is up and running inside Docker

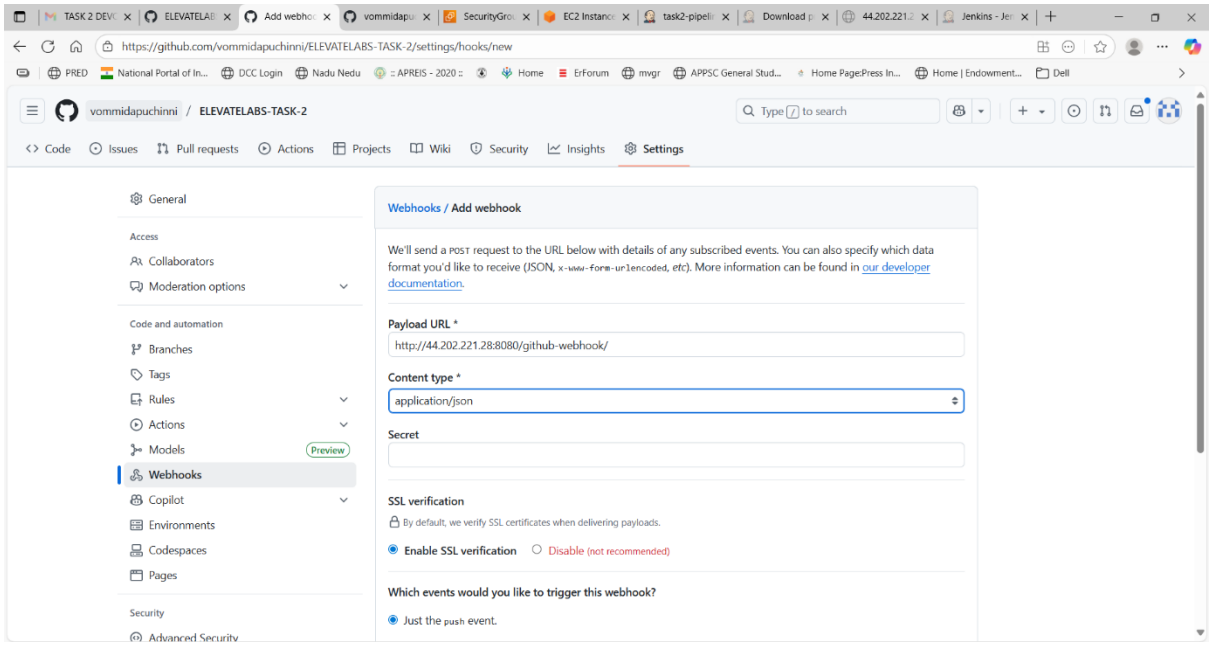


I am going to add a webhook for automatic triggering when there is an change in git repo

9. GitHub Webhook Setup

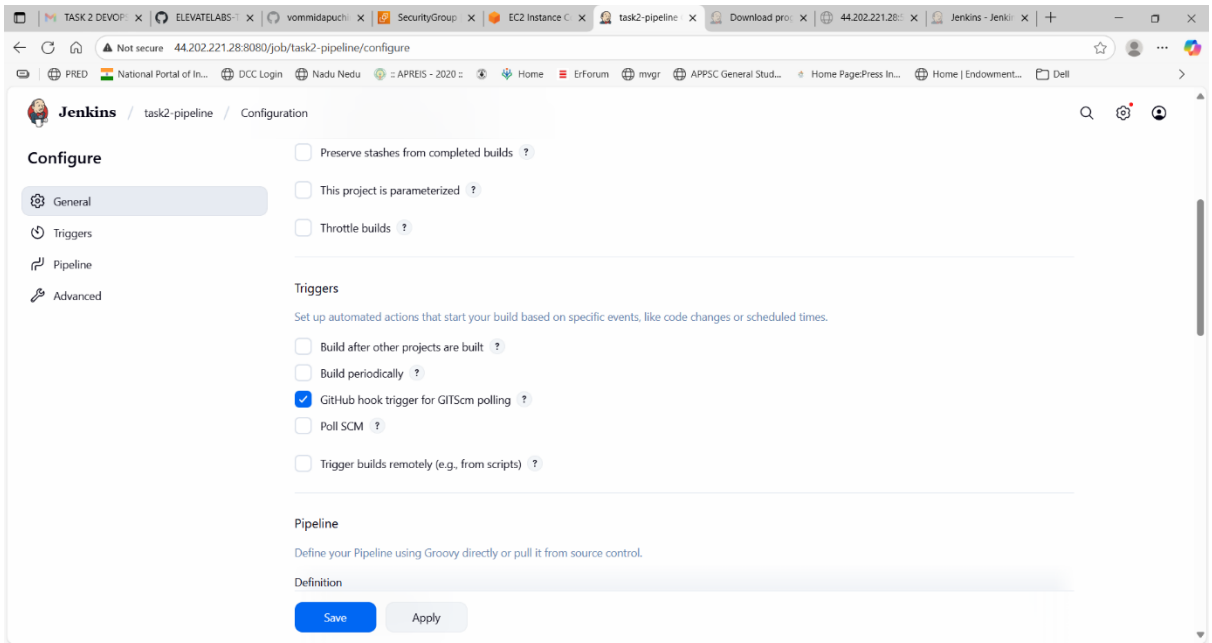
- GitHub → Repo → Settings → Webhooks → Add Webhook
- Payload URL: http://<your-ec2-ip>:8080/github-webhook/
- Trigger: Just push events

- Click add webhook



10. Enable Webhook Trigger in Jenkins Job

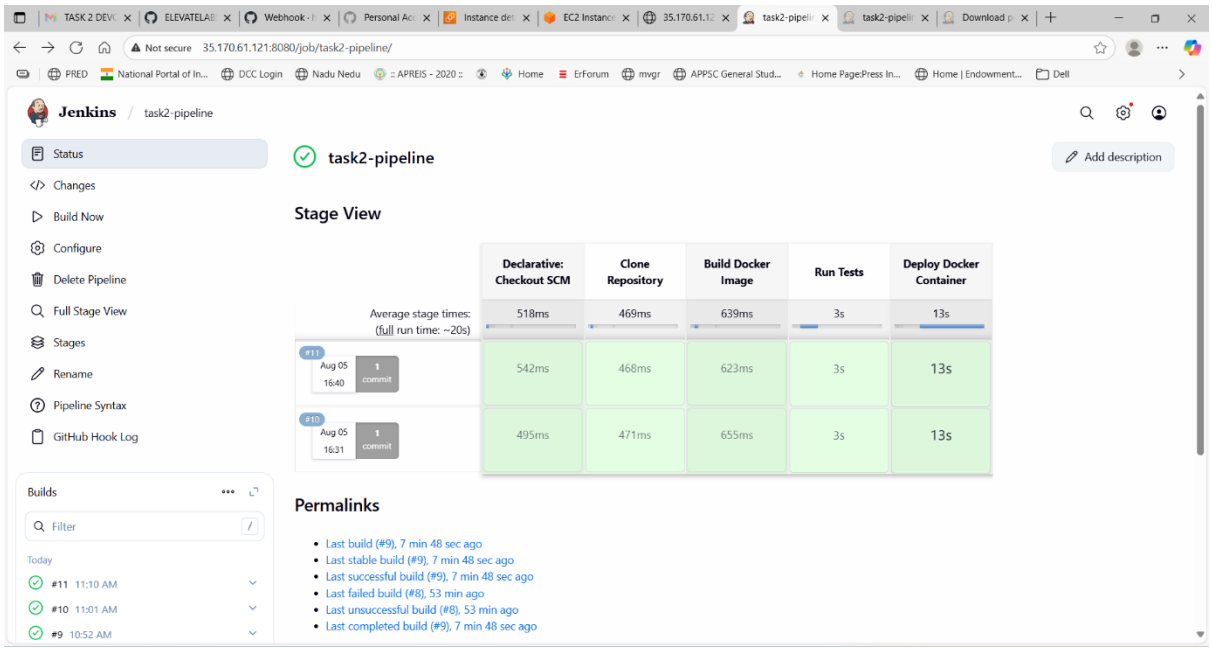
- Go to your Jenkins job → Configure → under Build Triggers:
- Tick: GitHub hook trigger for GITScm polling
- Save it come back



11. Pipeline Triggered Automatically

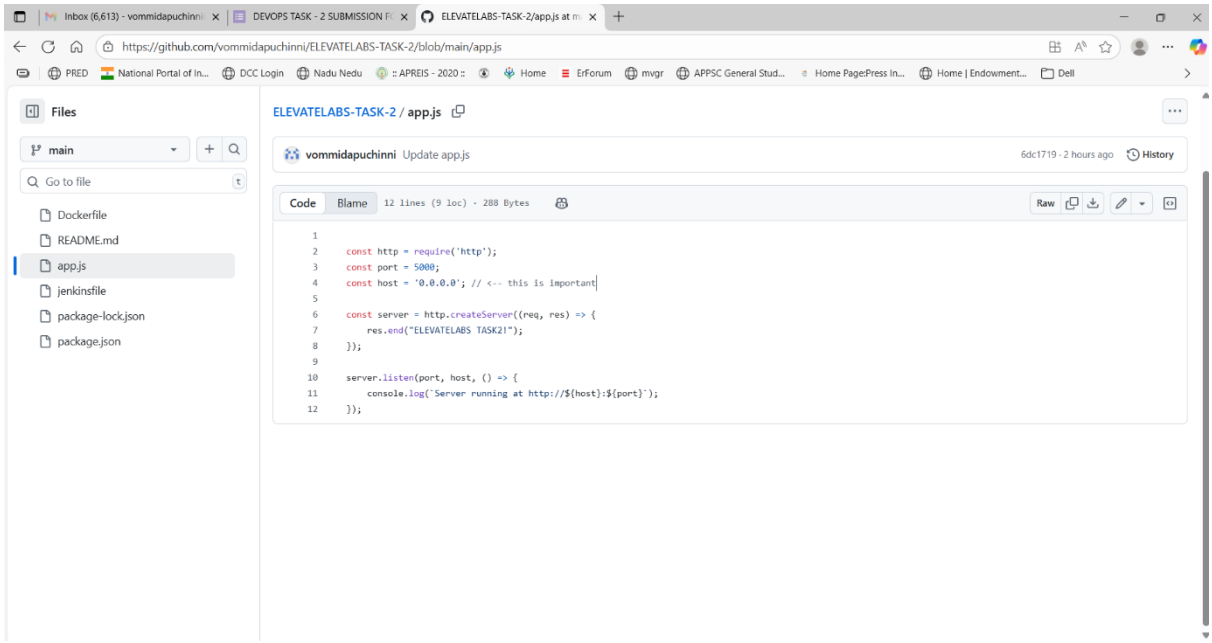
- Every time I push code to GitHub, Jenkins automatically:
  - Clones the repo
  - Builds the Docker image
  - Runs the container

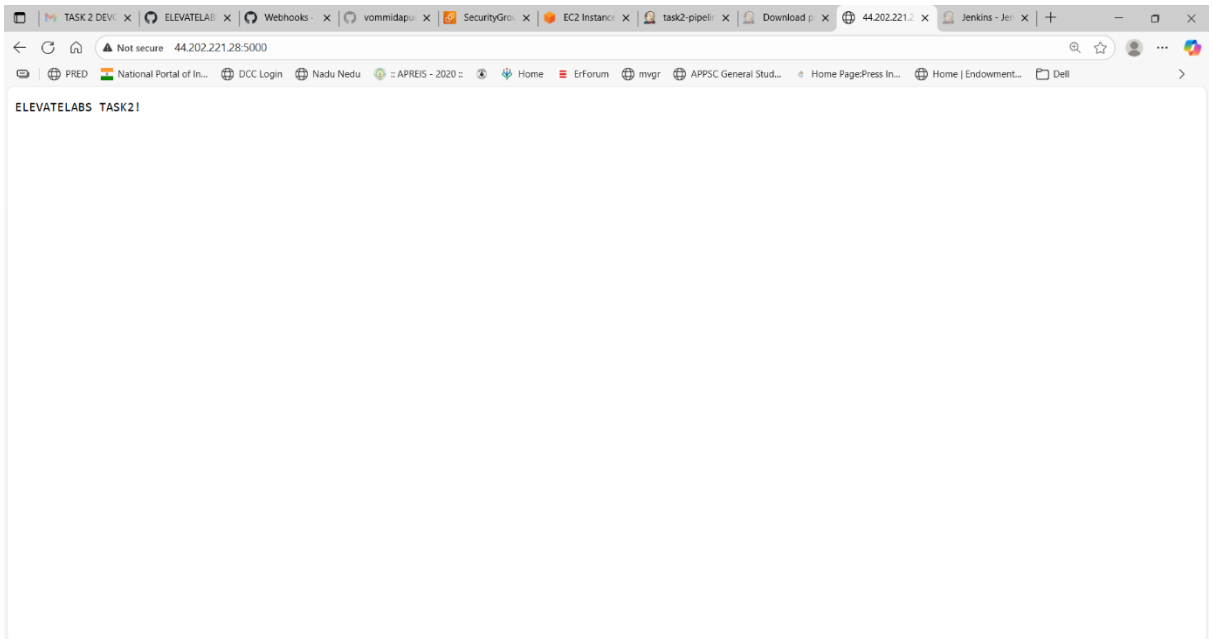
12. Here as mentioned in instructions I have build tested and deployed.



Checking that automatic triggering is happening or not I have changed the code in app.js (node.js app deployed via Jenkins to elevatelabs task2!)

Its automatically triggered and accessed the application via public ip





13. Troubleshooting

Issue: Jenkins can't run Docker commands
Fix: Add Jenkins user to Docker group

sudo usermod -aG docker jenkins

sudo systemctl restart docker

Issue: App not accessible on browser
Fix:

- Check EC2 security group has port 5000 open
- Confirm container is running with docker ps
- Check logs using docker logs myapp

Issue: Webhook not triggering build
Fix:

- Ensure webhook URL is correct
- Jenkins GitHub plugin installed
- GitHub and Jenkins connected properly

14. Conclusion

This project helped me learn how to:

- Create and deploy a Node.js app
- Use Jenkins to automate CI/CD pipeline
- Containerize an app using Docker
- Host and manage the pipeline on AWS EC2
- Integrate GitHub with Jenkins using webhooks

I also solved errors related to Docker permissions, webhook misconfiguration, and deployment issues during the task.