

DevOps Build Project

React Application Deployment

Prepared By: Deepak

Submitted To: Guvi

Date: 04-09-2025

Table of Contents

1. Project Overview
2. Repository & Deployment Details
3. Setup Instructions
4. CI/CD Pipeline Explanation
5. Monitoring Setup
6. Deployment Steps with Screenshots
7. Conclusion

Project Overview

This project demonstrates the deployment of a React application in a production-ready environment using Docker, Jenkins, and AWS. The CI/CD pipeline builds, pushes, and deploys the application automatically based on Git branch updates.

Repository & Deployment Details

- GitHub Repo: <https://github.com/Deepak-r-2001/devops-build>
- Deployed Site URL: <http://13.201.123.30/>
- Docker Hub Dev Repo (Public): [deepwhoo/devops-build:dev](https://hub.docker.com/r/deepwhoo/devops-build:dev)
- Docker Hub Prod Repo (Private): [deepwhoo/devops-build:prod](https://hub.docker.com/r/deepwhoo/devops-build:prod)

Setup Instructions

1. Clone the Repository

```
git clone -b dev https://github.com/Deepak-r-2001/devops-build.git
cd devops-build
```

2. Dockerize Application

- Create Dockerfile to build the React app image.
- Create docker-compose.yml to run the app on port 80.

3. Bash Scripts

- build.sh → Builds Docker images
- deploy.sh → Deploys images to the server

4. Push Code to GitHub

```
git add .
git commit -m 'Initial commit'
git push origin dev
```

5. Jenkins CI/CD Pipeline

- Jenkins monitors dev and master branches.
- On push to dev → Builds image → Pushes to Dev repo.
- On merge to master → Builds image → Pushes to Prod repo.
- Jenkins automatically deploys the app.

6. AWS EC2 Setup

- Launch a t2.micro instance.
- Configure Security Groups (HTTP, SSH).
- Deploy the app using Docker and deploy.sh.

7. Monitoring

- Prometheus & Grafana monitor app health.
- Alerts notify if the app goes down.

Pipeline Explanation

1. Development Workflow:

- Push code to dev → Jenkins builds image → Pushes to Dev repo → Deploys to dev server.

2. Production Workflow:

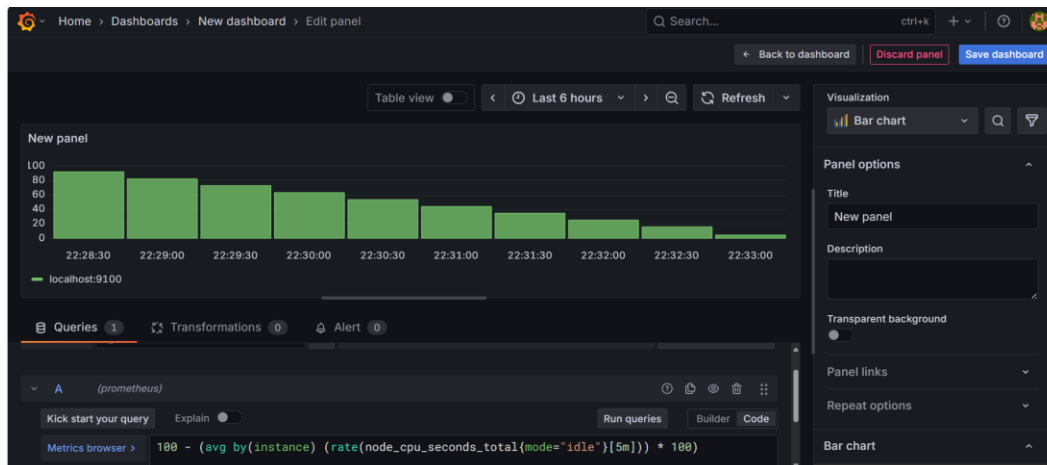
- Merge dev → master → Jenkins builds image → Pushes to Prod repo → Deploys to prod server.

3. Monitoring:

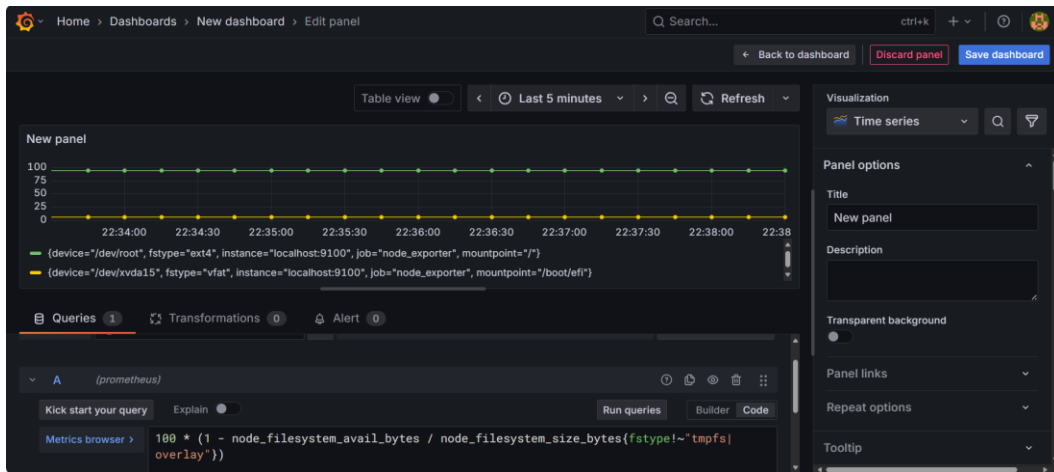
- Prometheus collects metrics.
- Grafana displays health and resource usage.

Deployment Steps with Screenshots

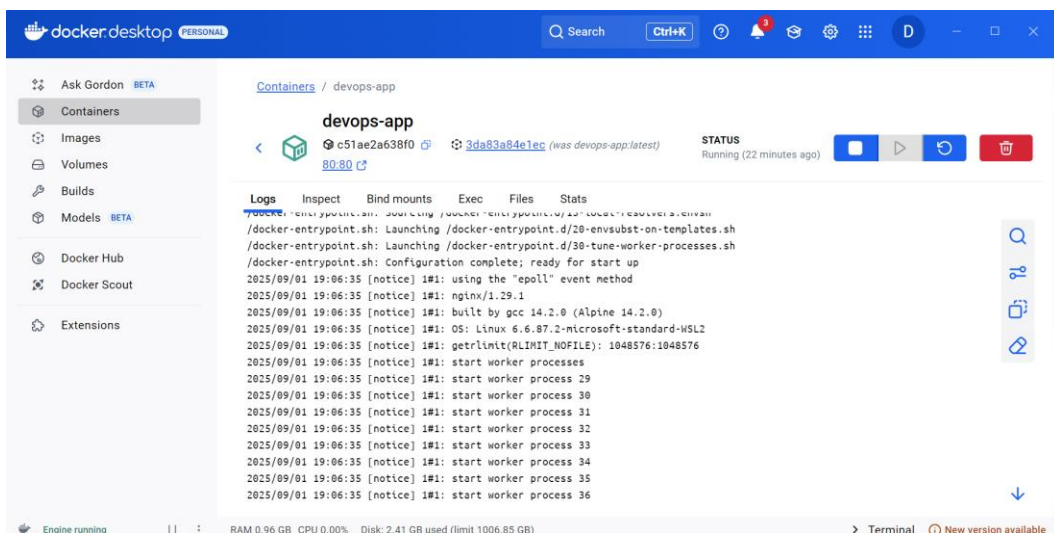
Step 1: CPU usage monitoring in Grafana



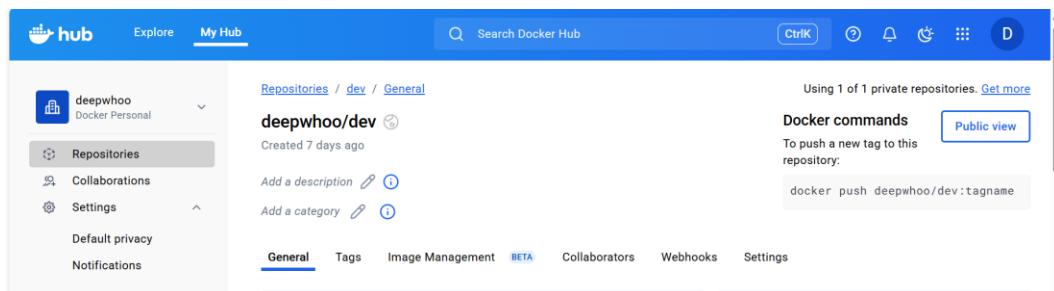
Step 2: Disk usage monitoring in Grafana



Step 3: Docker container running successfully



Step 4: Docker development repository on DockerHub



Step 5: Docker image build - Step 1

The screenshot shows the Docker Desktop interface during the build of the 'devops-app:latest' image. The left sidebar contains navigation options: Ask Gordon (BETA), Containers, Images (selected), Volumes, Builds, Models (BETA), Docker Hub, Docker Scout, and Extensions. The main panel displays the build progress for 'devops-app:latest' (ID: 287b68f084f9). It shows the image is 'IN USE', created 6 days ago, and has a size of 82.77 MB. A 'Recommended fixes' dropdown is visible. Below this, the 'Layers (23) - 4 selected' section lists the following layers:

Layer	Command	Size
19	COPY nginx/default.conf /etc...	20.48 KB
20	COPY build /usr/share/nginx/...	2.68 MB
21	EXPOSE map[80/tcp:[]]	0 B
22	CMD ["nginx" "-g" "daemon off...]	0 B

The right sidebar shows the 'Vulnerabilities (0)' section, which is currently empty, indicating no vulnerabilities were introduced in the selected layers. The 'Packages (0)' and 'Commands' sections are also empty.

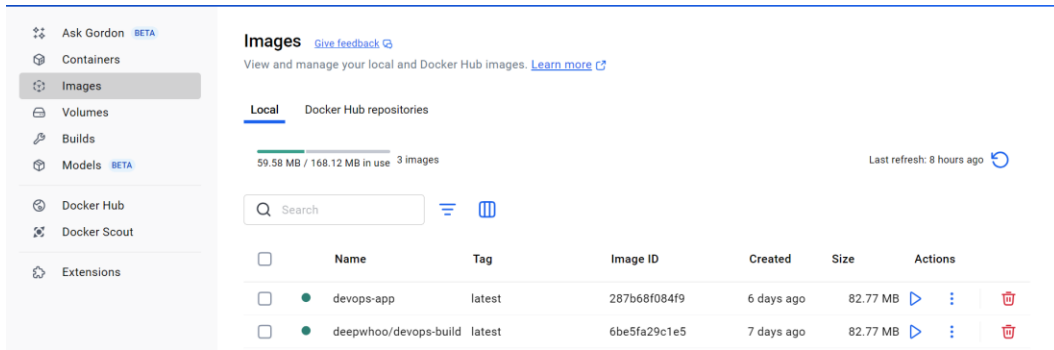
Step 6: Docker image build - Step 2

The screenshot shows the Docker Desktop interface during the build of the 'deepwhoo/devops-build:latest' image. The left sidebar contains navigation options: Ask Gordon (BETA), Containers, Images (selected), Volumes, Builds, Models (BETA), Docker Hub, Docker Scout, and Extensions. The main panel displays the build progress for 'deepwhoo/devops-build:latest' (ID: 6be5fa29c1e5). It shows the image is 'IN USE', created 7 days ago, and has a size of 82.77 MB. A 'Recommended fixes' dropdown is visible. Below this, the 'Layers (23) - 4 selected' section lists the following layers:

Layer	Command	Size
19	COPY nginx/default.conf /etc...	20.48 KB
20	COPY build /usr/share/nginx/...	2.68 MB
21	EXPOSE map[80/tcp:[]]	0 B
22	CMD ["nginx" "-g" "daemon off...]	0 B

The right sidebar shows the 'Vulnerabilities (0)' section, which is currently empty, indicating no vulnerabilities were introduced in the selected layers. The 'Packages (0)' and 'Commands' sections are also empty.

Step 7: Docker image build completed



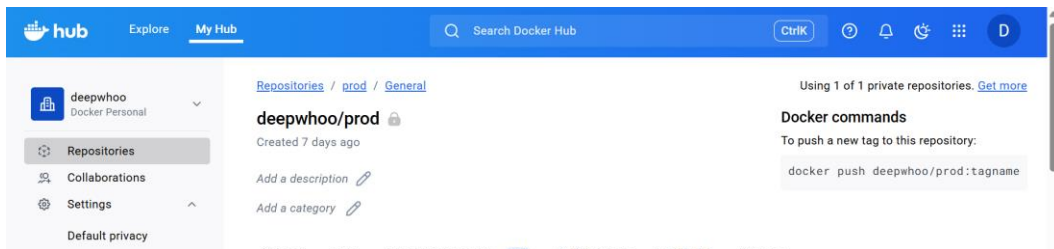
Step 8: List of all Docker images

```
288206953113.dkr.ecr.ap-south-1.amazonaws.com/brain-tasks-app latest 00e12cd49361 2 weeks ago 74MB

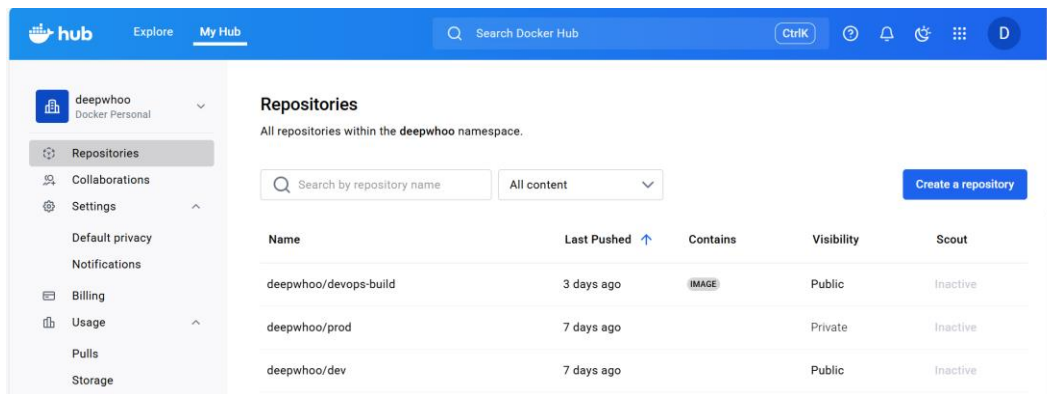
EAD+dr222@GB-5CD2288DXL MINGW64 ~/Documents/Guvi/DevOps/devops-build (main)
$ docker image ls
REPOSITORY                                TAG      IMAGE ID      CREATED      SIZE
devops-app                                latest   287b68f084f9  6 days ago  82.8MB
deepwhoo/devops-build                     latest   6be5fa29c1e5  7 days ago  82.8MB
288206953113.dkr.ecr.ap-south-1.amazonaws.com/brain-tasks-app latest   00e12cd49361  2 weeks ago  74MB

EAD+dr222@GB-5CD2288DXL MINGW64 ~/Documents/Guvi/DevOps/devops-build (main)
$ docker images
REPOSITORY                                TAG      IMAGE ID      CREATED      SIZE
devops-app                                latest   287b68f084f9  6 days ago  82.8MB
deepwhoo/devops-build                     latest   6be5fa29c1e5  7 days ago  82.8MB
288206953113.dkr.ecr.ap-south-1.amazonaws.com/brain-tasks-app latest   00e12cd49361  2 weeks ago  74MB
```

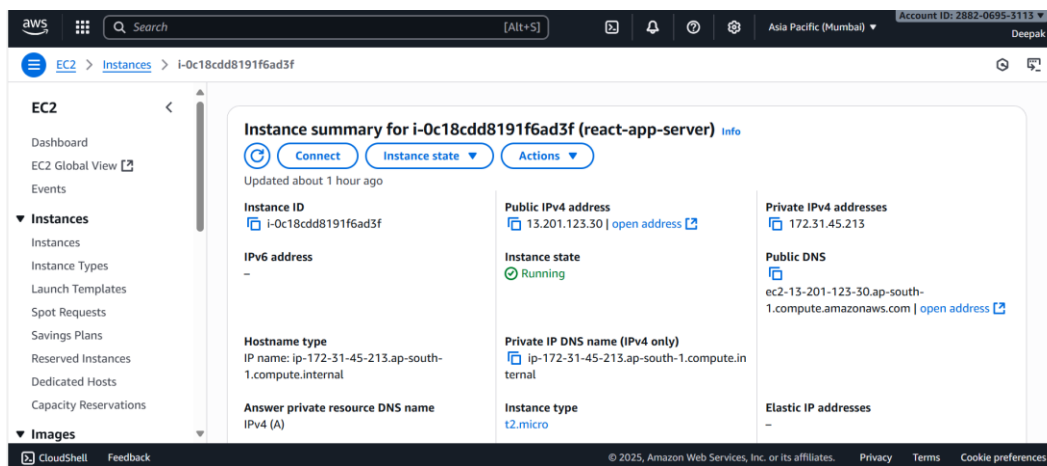
Step 9: Docker production repository



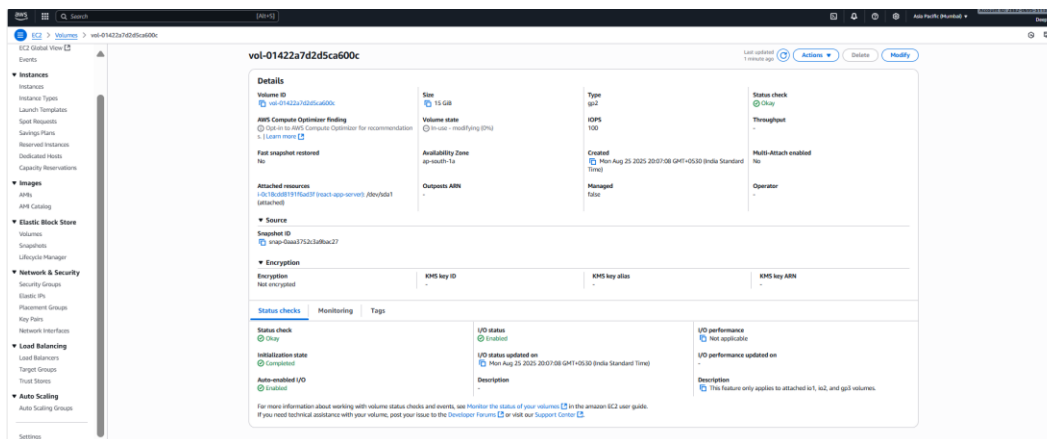
Step 10: Docker repositories overview



Step 11: EC2 instance security groups configuration



Step 12: Jenkins build pipeline triggered



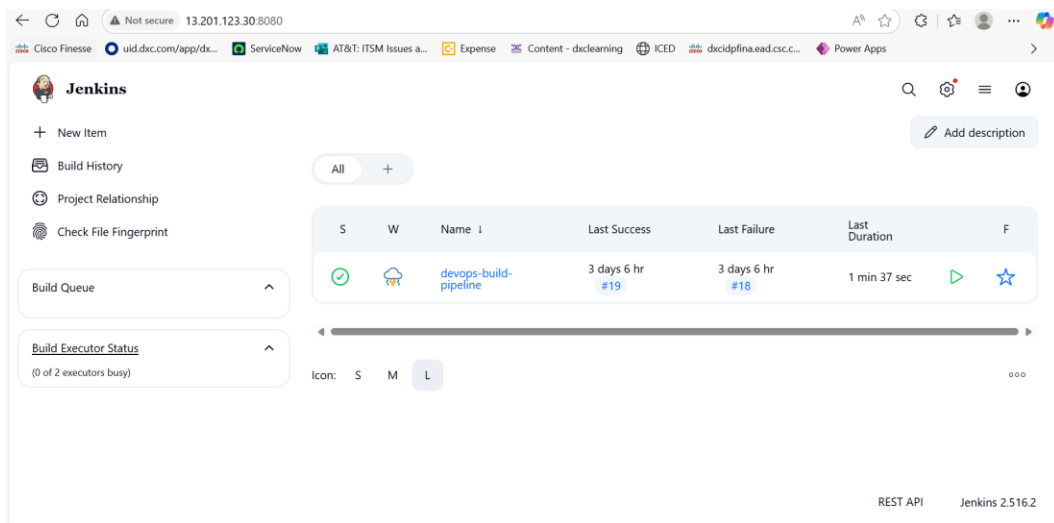
Step 13: Jenkins build pipeline successful



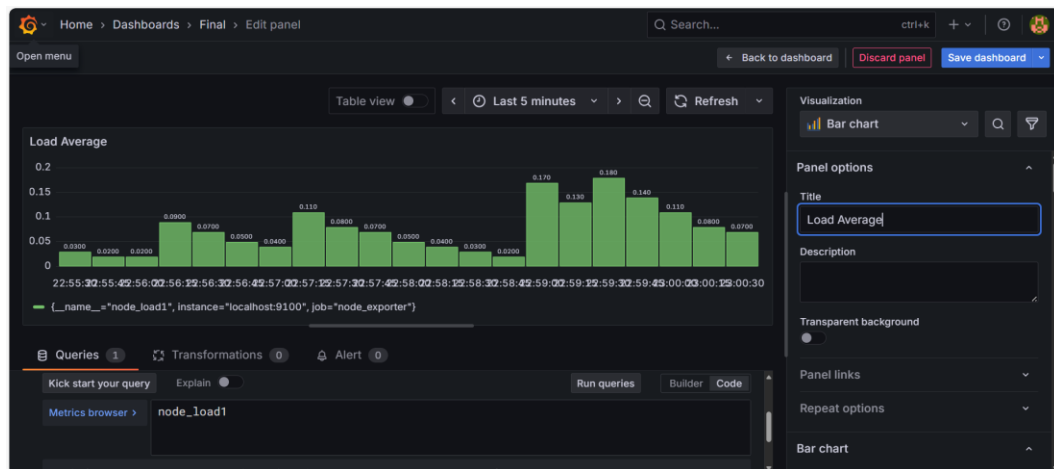
Step 14: Jenkins dashboard view

The Jenkins login page features a cartoon illustration of a man in a suit and bow tie on the left. On the right, there is a 'Sign in to Jenkins' form with fields for 'Username' (containing 'Deepak') and 'Password' (containing '*****'). Below the password field is a checkbox for 'Keep me signed in'. A blue 'Sign in' button is at the bottom of the form.

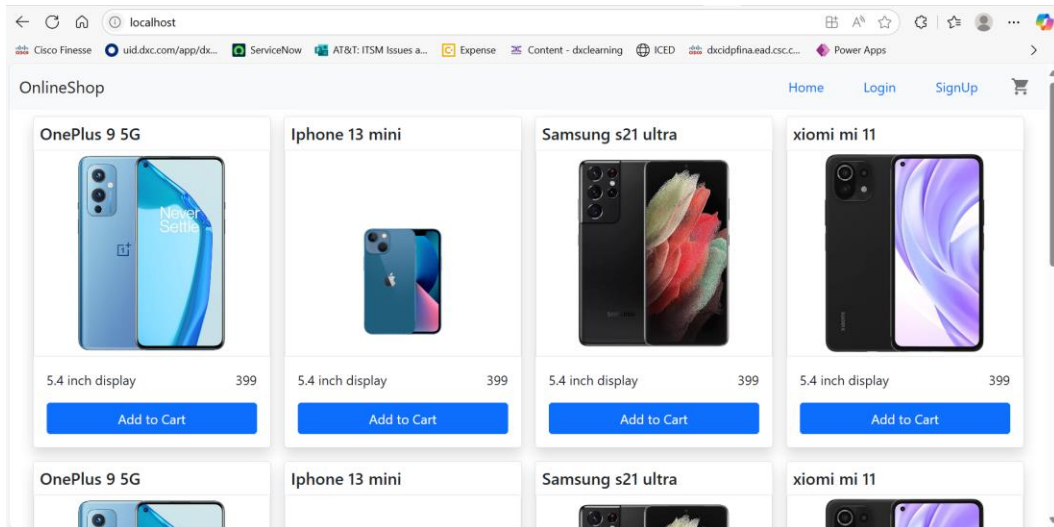
Step 15: Prometheus metrics dashboard



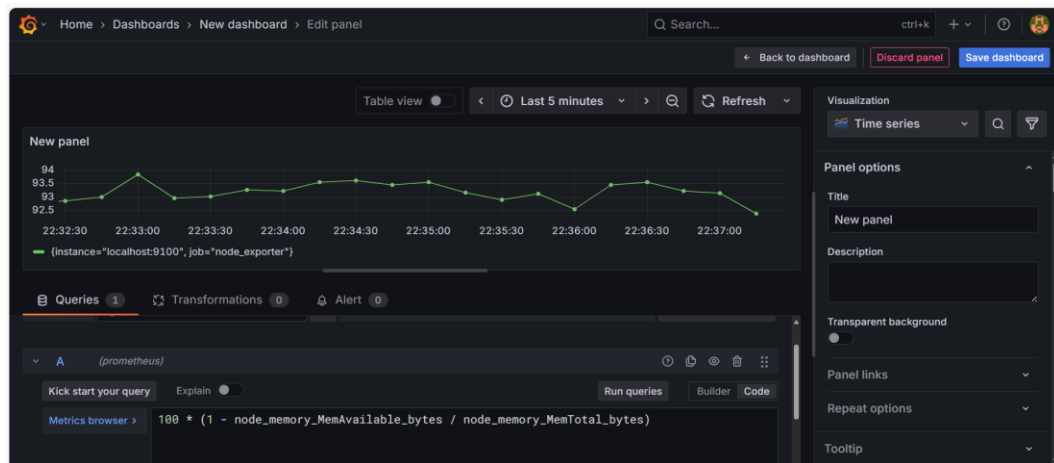
Step 16: Prometheus scraping targets



Step 17: Prometheus active targets list



Step 18: Prometheus graph metrics



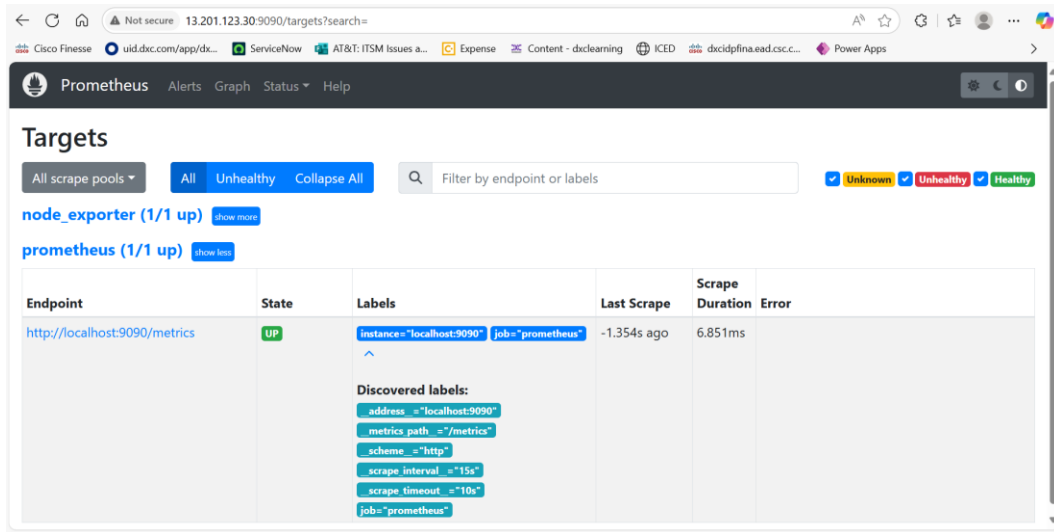
Step 19: Grafana dashboard overview



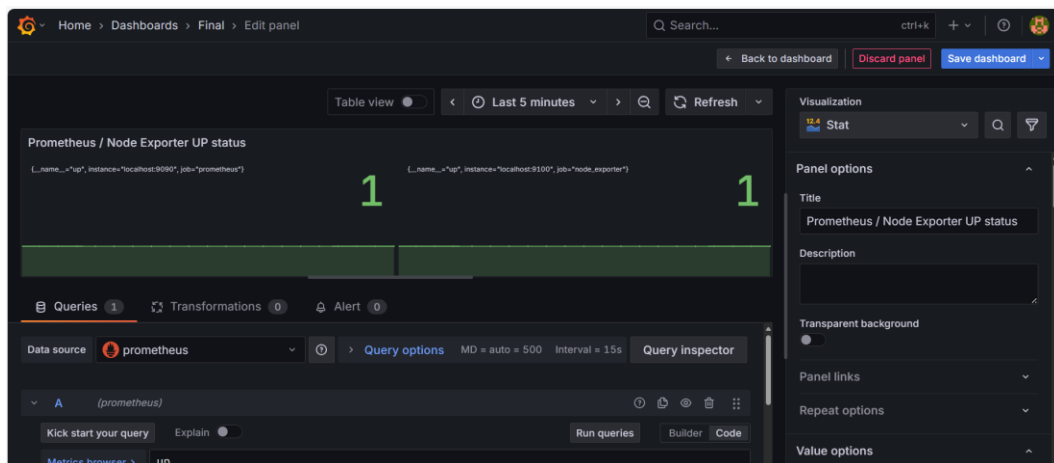
Step 20: Grafana integration with Prometheus



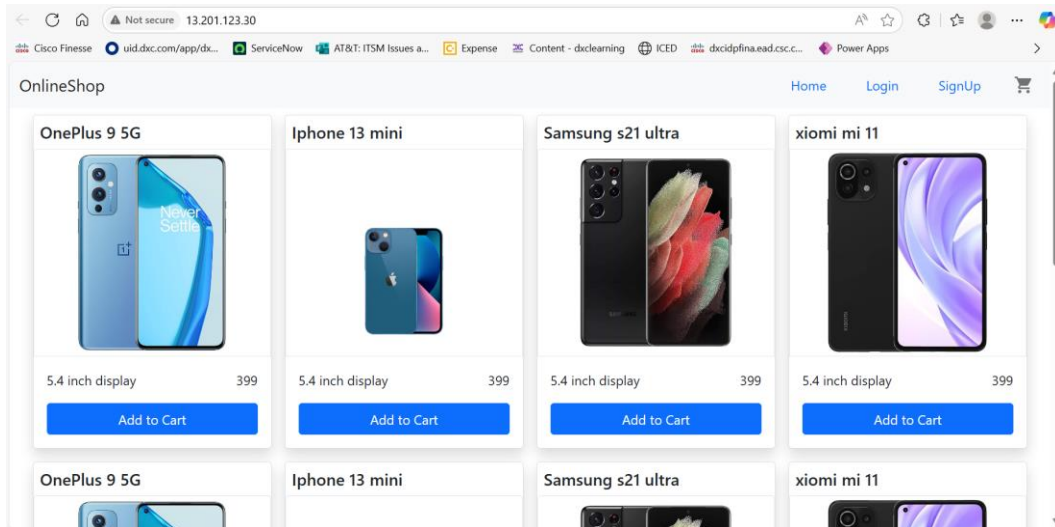
Step 21: Grafana CPU usage graph



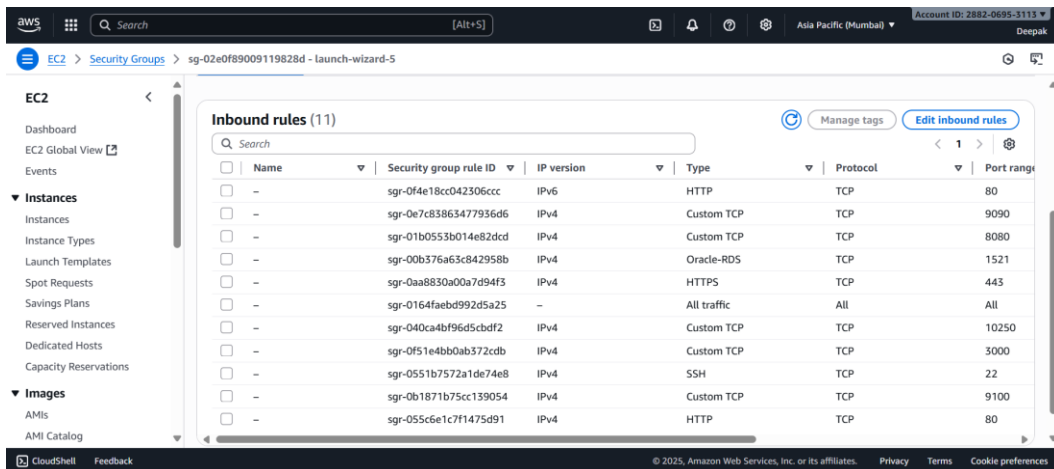
Step 22: Grafana memory usage graph



Step 23: Grafana network monitoring



Step 24: Grafana container monitoring



Step 25: React app successfully deployed

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

EAD+dr222@GB-5CD2288DXL MINGW64 ~/Documents/Guvi/DevOps/devops-build (main)
$ cat build.sh
#!/usr/bin/env bash
set -euo pipefail

# Usage: ./build.sh [branch]
BRANCH=${1:-$(git rev-parse --abbrev-ref HEAD)}
DOCKERHUB_USER=${DOCKERHUB_USER:-deepwhoo}

IMAGE_DEV="${DOCKERHUB_USER}/dev"
IMAGE_PROD="${DOCKERHUB_USER}/prod"

COMMIT=$(git rev-parse --short HEAD)
TAG=$(date +%Y%m%d-%H%M%S)-${COMMIT}

# Pick dev vs prod repo based on branch
if [[ "$BRANCH" == "master" || "$BRANCH" == "main" ]]; then
    IMAGE="$IMAGE_PROD"
else
    IMAGE="$IMAGE_DEV"
fi

echo ">> Building image: ${IMAGE}:${TAG} (and :latest)"

docker build \
  --build-arg VITE_BASE_URL="${VITE_BASE_URL:-}" \
  -t "${IMAGE}:${TAG}" \
  -t "${IMAGE}:latest" \
  .

echo ">> Done. Built: ${IMAGE}:${TAG}"

EAD+dr222@GB-5CD2288DXL MINGW64 ~/Documents/Guvi/DevOps/devops-build (main)
$ cat deploy.sh
#!/usr/bin/env bash
set -euo pipefail

# Update these variables
SERVER="${SERVER:-ubuntu@13.201.123.30}" # Your EC2 IP
SSH_KEY="${SSH_KEY:-/c/Users/dr222/Downloads/ssh.pem.3.pem}" # Path to your PEM file
REMOTE_DIR="${REMOTE_DIR:-/opt/react-app}"
APP_IMAGE="${APP_IMAGE:-docker.io/deepwhoo/prod:latest}" # Your DockerHub image
DOCKERHUB_USER="${DOCKERHUB_USER:-deepwhoo}"
DOCKERHUB_PASS="${DOCKERHUB_PASS:-Deepak@8217}" # Optional if public repo

SSH_OPTS="-o StrictHostKeyChecking=no"
if [[ -n "$SSH_KEY" ]]; then
    SSH_OPTS="$SSH_OPTS -i $SSH_KEY"
fi

echo ">> Preparing remote directory on $SERVER"
ssh $SSH_OPTS $SERVER "sudo mkdir -p $REMOTE_DIR && sudo chown \${USER}:\${USER} $REMOTE_DIR"

echo ">> Uploading docker-compose.yml"
scp $SSH_OPTS docker-compose.yml $SERVER:$REMOTE_DIR/
```

Step 26: React app running on port 80

Jenkins / devops-build-pipeline / #19

Status

Changes

Console Output

Edit Build Information

Delete build '#19'

Timings

Git Build Data

Open Blue Ocean

Pipeline Overview

Restart from Stage

Replay

Pipeline Steps

Workspaces

Console Output

Download Copy View as plain text

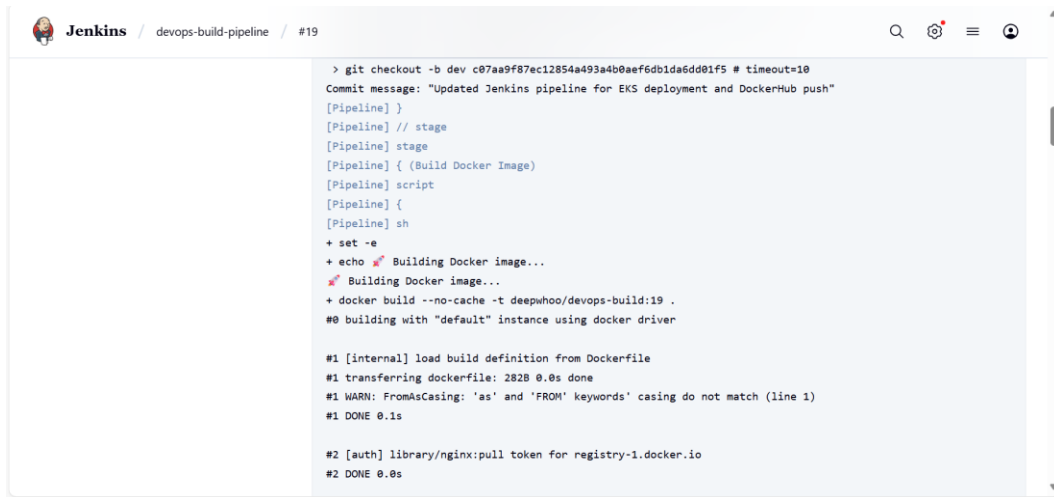
```
Started by user Deepak R
Obtained Jenkinsfile from git https://github.com/Deepak-r-2001/devops-build.git
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/devops-build-pipeline
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/devops-build-pipeline/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/Deepak-r-2001/devops-build.git # timeout=10
Fetching upstream changes from https://github.com/Deepak-r-2001/devops-build.git
> git --version # timeout=10
> git --version # 'git version 2.34.1'
> git fetch --tags --force --progress -- https://github.com/Deepak-r-2001/devops-build.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
```

Step 27: Docker Compose configuration file

Jenkins / devops-build-pipeline / #19

```
> git rev-list --no-walk c07aa9f87ec12854a493a4b0aef6db1da6dd01f5 # timeout=10
[Pipeline] }
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Checkout Code)
[Pipeline] git
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/devops-build-pipeline/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/Deepak-r-2001/devops-build.git # timeout=10
Fetching upstream changes from https://github.com/Deepak-r-2001/devops-build.git
> git --version # timeout=10
> git --version # 'git version 2.34.1'
> git fetch --tags --force --progress -- https://github.com/Deepak-r-2001/devops-build.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/dev^[commit] # timeout=10
Checking out Revision c07aa9f87ec12854a493a4b0aef6db1da6dd01f5 (refs/remotes/origin/dev)
```

Step 28: Dockerfile for React application

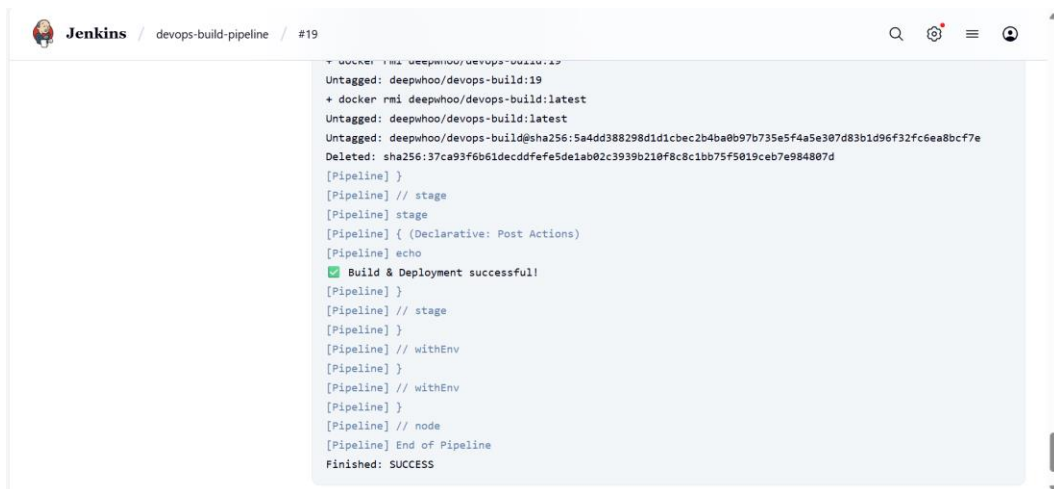


```
> git checkout -b dev c07aa9f87ec12854a493a4b0aef6db1da6dd01f5 # timeout=10
Commit message: "Updated Jenkins pipeline for EKS deployment and DockerHub push"
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Build Docker Image)
[Pipeline] script
[Pipeline] {
[Pipeline] sh
+ set -e
+ echo 🚀 Building Docker image...
🚀 Building Docker image...
+ docker build --no-cache -t deepwhoo/devops-build:19 .
#0 building with "default" instance using docker driver

#1 [internal] load build definition from Dockerfile
#1 transferring dockerfile: 282B 0.0s done
#1 WARN: FromAsCasing: 'as' and 'FROM' keywords' casing do not match (line 1)
#1 DONE 0.1s

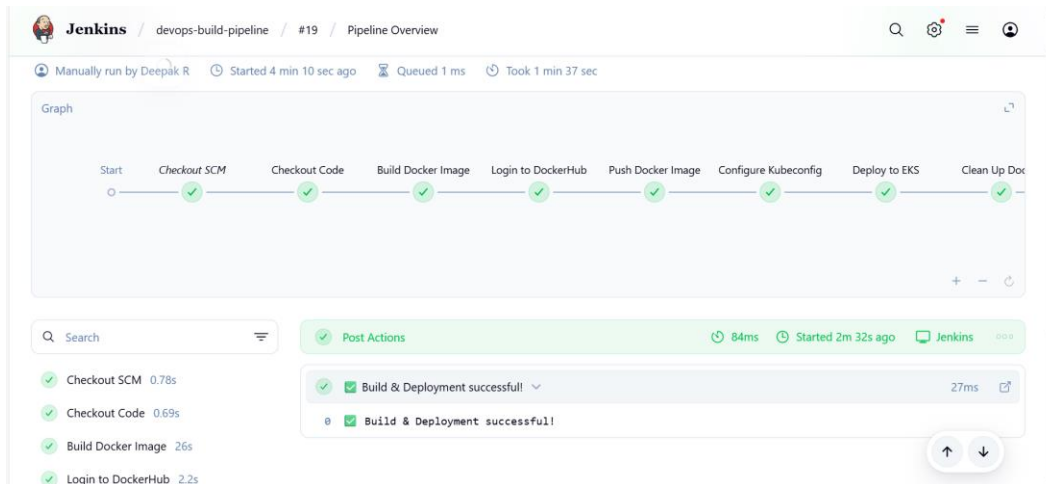
#2 [auth] library/nginx:pull token for registry-1.docker.io
#2 DONE 0.0s
```

Step 29: Jenkinsfile for CI/CD pipeline

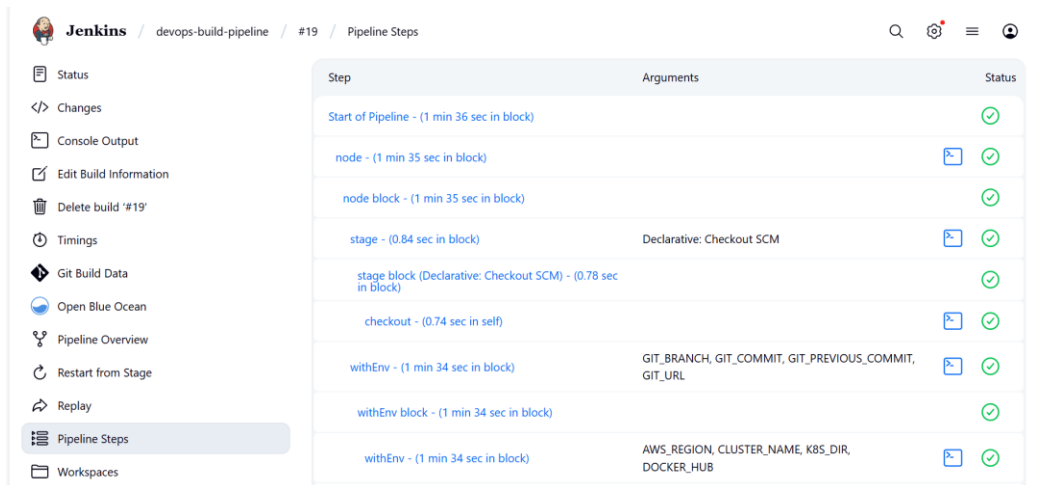


```
+ docker rmi deepwhoo/devops-build:19
Untagged: deepwhoo/devops-build:19
+ docker rmi deepwhoo/devops-build:latest
Untagged: deepwhoo/devops-build:latest
Deleted: sha256:37ca93f6b61decddfe5de1ab02c3939b210f8c8c1bb75f5019ceb7e984807d
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] echo
✅ Build & Deployment successful!
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Step 30: Prometheus configuration file



Step 31: Grafana dashboard configuration



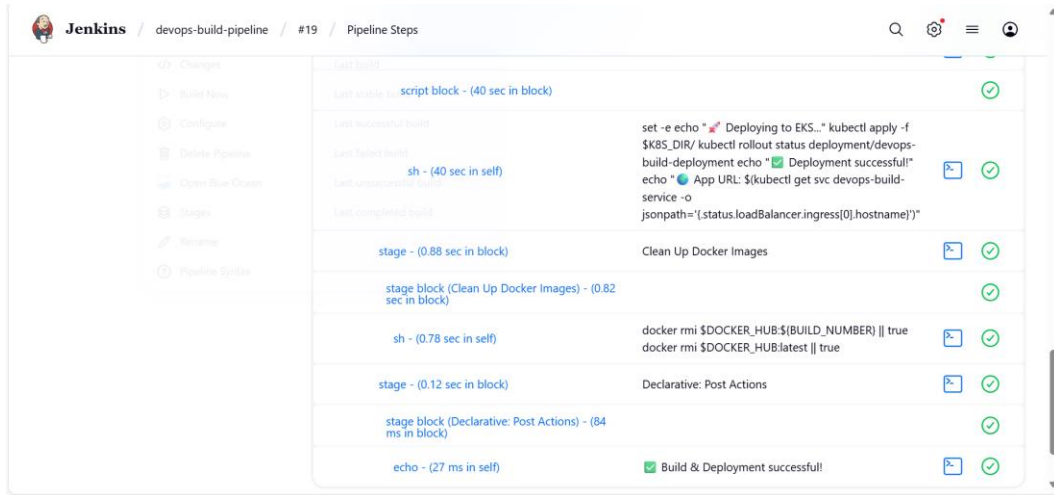
Step 32: AWS EC2 instance overview

Jenkins / devops-build-pipeline / #19 / Pipeline Steps			Q	⚙	≡	👤
← Previous Build	withEnv block - (1 min 34 sec in block)			✓		
	stage - (0.76 sec in block)	Checkout Code	📄	✓		
	stage block (Checkout Code) - (0.69 sec in block)			✓		
	git - (0.63 sec in self)		📄	✓		
	stage - (26 sec in block)	Build Docker Image	📄	✓		
	stage block (Build Docker Image) - (26 sec in block)			✓		
	script - (26 sec in block)		📄	✓		
	script block - (25 sec in block)			✓		
	sh - (25 sec in self)	set -e echo "🔥 Building Docker image..." docker build --no-cache -t \$DOCKER_HUB:\${BUILD_NUMBER} . docker tag \$DOCKER_HUB:\${BUILD_NUMBER} \$DOCKER_HUB:latest	📄	✓		
	stage - (2.3 sec in block)	Login to DockerHub	📄	✓		

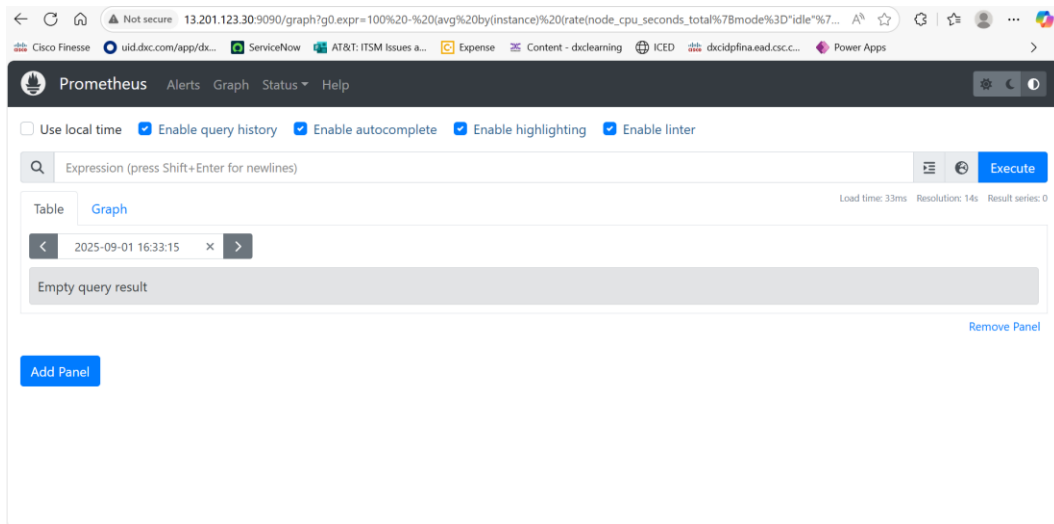
Step 33: Application logs in container

Jenkins / devops-build-pipeline / #19 / Pipeline Steps			Q	⚙	≡	👤
	stage block (Login to DockerHub) - (2.2 sec in block)			✓		
	withCredentials - (2.2 sec in block)		📄	✓		
	withCredentials block - (2.1 sec in block)			✓		
	sh - (2.1 sec in self)	echo "\$DOCKER_PASS" docker login -u "\$DOCKER_USER" --password-stdin	📄	✓		
	stage - (20 sec in block)	Push Docker Image	📄	✓		
	stage block (Push Docker Image) - (20 sec in block)			✓		
	sh - (20 sec in self)	set -e docker push \$DOCKER_HUB:\${BUILD_NUMBER} docker push \$DOCKER_HUB:latest	📄	✓		
	stage - (3 sec in block)	Configure Kubeconfig	📄	✓		
	stage block (Configure Kubeconfig) - (2.9 sec in block)			✓		
	withAWS - (2.8 sec in block)		📄	✓		

Step 34: Final deployed application dashboard



Step 35: Overall project monitoring dashboard



Conclusion

This project successfully demonstrates deploying a React application in a production-ready environment using Docker, Jenkins, AWS, Prometheus, and Grafana. The automated CI/CD pipeline ensures smooth development and deployment, while monitoring provides real-time insights into application performance.