Deploying Applications on Docker and Kubernetes with Jenkins CI/CD pipeline using AWS and DevSecOps Practices

Project by

Rajmane Amit Shivprasad	**(230944223030)**
Piyush Arun Patil	**(230944223024)**
Gaurav Nivas Jadhav	**(230944223012)**
Raut Kunal Sudam	**(230944223032)**

Under the guidance of

Mr. Sandeep Walvekar

Sunbeam Institute of Information Technology,

Pune (Maharashtra)

PG-DITISS -2024

Contents

Introduction

1.1 Introduction

- In today's rapidly evolving technological landscape, the deployment of applications demands agility, scalability, and security.
- Traditional deployment methods often struggle to keep pace with the dynamic nature of modern software development, leading to inefficiencies and security vulnerabilities.
- Organizations are increasingly turning to containerization and orchestration solutions such as Docker and Kubernetes, coupled with Continuous Integration/Continuous Delivery (CI/CD) pipelines powered by tools like Jenkins.
- Integration of DevSecOps practices into the deployment process has become imperative to ensure robust security measures throughout the software development lifecycle.

Introduction (contd..)

1.2 Motivation

- To bridge the gap between agility, scalability, and security in application deployment.
- To streamline the deployment process of containerized applications while ensuring robust security measures from inception to production.
- Automate deployment, continuous integration, delivery, and security scanning processes.
- Ensure that security remains a top priority throughout the development process, ultimately leading to the delivery of high-quality, secure software to end-users.

Introduction (contd..)

1.3 Problem Statement

- There are inefficiencies and security vulnerabilities in traditional deployment methods. Difficulty in keeping pace with modern software development dynamics. Organizations require a solution that bridges agility, scalability, and security in application deployment.
- We are simplifying how we deploy software while making sure it's very secure from the beginning until it's in use is really important. It helps us stay competitive and lowers the chances of security problems or breaking rules.

Introduction (contd..)

1.4 Objectives

- To Automate deployment, continuous integration, delivery, and security scanning processes.
- Promote a culture of collaboration, agility, and security across the software development lifecycle.
- Ensure that security remains a top priority throughout the development process, ultimately leading to the delivery of high-quality, secure software to end-users.

Literature Survey

Sr.No.	Paper Title	Authors
01	Container orchestration and its implications on enterprise networking	Mohanty, B., & Sarma, S.
02	Continuous Integration and Continuous Delivery in the IT Industry: A Case Study	Stensholt, E., Landmark, A., & Fard, S. F.
03	DevSecOps: A Software Development Process for Securing Cloud-Native Applications	Chapman, C., Muthanna, A., & Rajani, S.

Proposed System

Our system architecture is designed to optimize the deployment and management of containerized applications while prioritizing security and scalability. Here's a breakdown of the components:

- **Amazon EC2 Instances:**
  - We utilize two instances: a t2.large for hosting Jenkins, OWASP Scan, Docker, Trivy, and the base operating system (Amazon Ubuntu AMI), and a t2.medium dedicated to Prometheus and Grafana for monitoring.
- **Amazon Elastic Kubernetes Service (EKS):**
  - EKS is employed for orchestrating and managing containerized applications. It offers scalability and reliability, automatically scaling based on workload demands.
- **CI/CD Pipeline with Jenkins:**
  - Jenkins acts as the CI/CD server, automating the build, test, and deployment processes. It seamlessly integrates with Docker and EKS for managing containerized deployments.

Proposed System (Contd.)

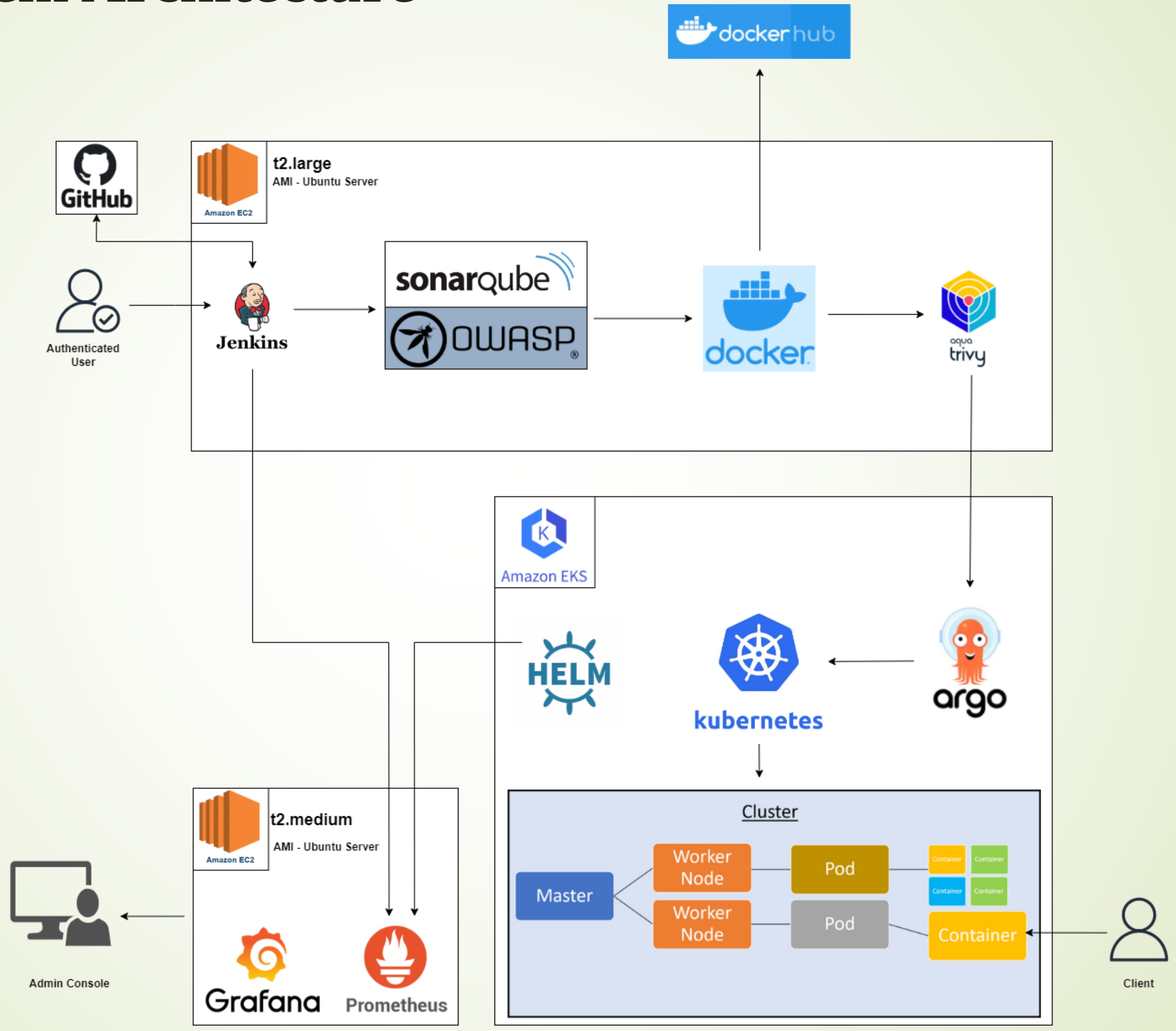
- ❑ **Source Code Management:**
- Git is a distributed version control system (VCS) designed to manage source code history and facilitate collaborative software development.
- ❑ **Docker:**
- Docker is an open-source platform that allows you to automate the deployment, scaling, and management of applications using containerization.
- ❑ **Package Management and Continuous Delivery:**
- **Helm** streamlines application deployment and management on Kubernetes.
- **ArgoCD** automates deployment based on Git repository changes, ensuring consistency and reliability through GitOps practices.

Proposed System (Contd.)

- **Security Scanning:**
  - **OWASP** Scan conducts security scans on applications deployed through Jenkins to identify vulnerabilities.
  - **Trivy** scans container images pre-deployment, ensuring secure deployments.
  - **SonarQube** is a free tool created by SonarSource that constantly checks the quality of code by automatically analyzing it for static analysis of code o detect bugs, code smells, and security vulnerabilities.
- **Monitoring with Prometheus and Grafana:**
  - Prometheus and Grafana collect and visualize metrics from EKS and other components for continuous monitoring and analysis, ensuring system health and performance optimization.

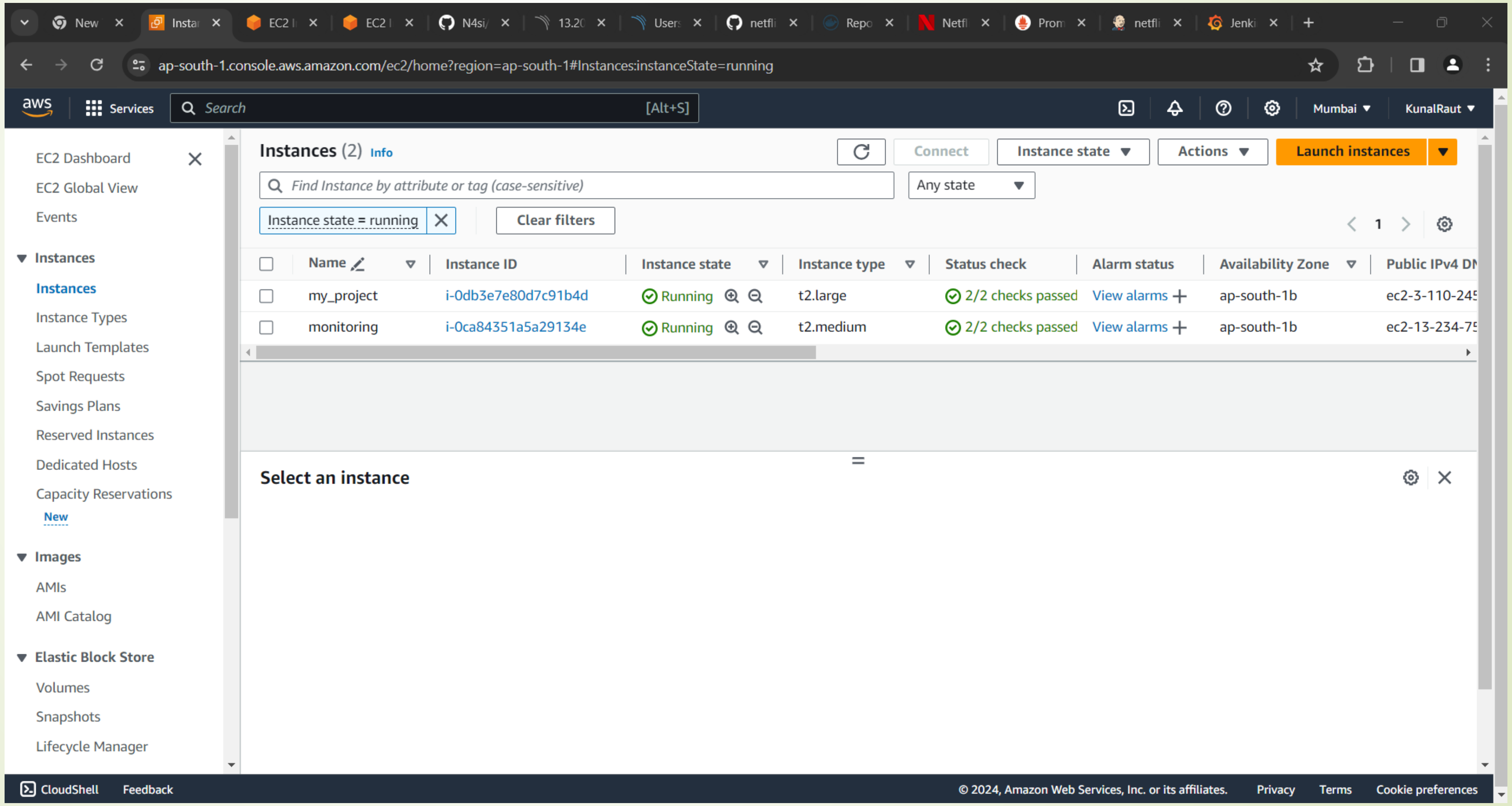
Overall, our system is designed to seamlessly integrate these components, ensuring efficient deployment processes, robust security measures, and continuous monitoring for scalability and availability.

System Architecture



Project Output

- AWS EC2 and EKS



ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#Instances:instanceState=running

Instances (2) Info

Instance state = running

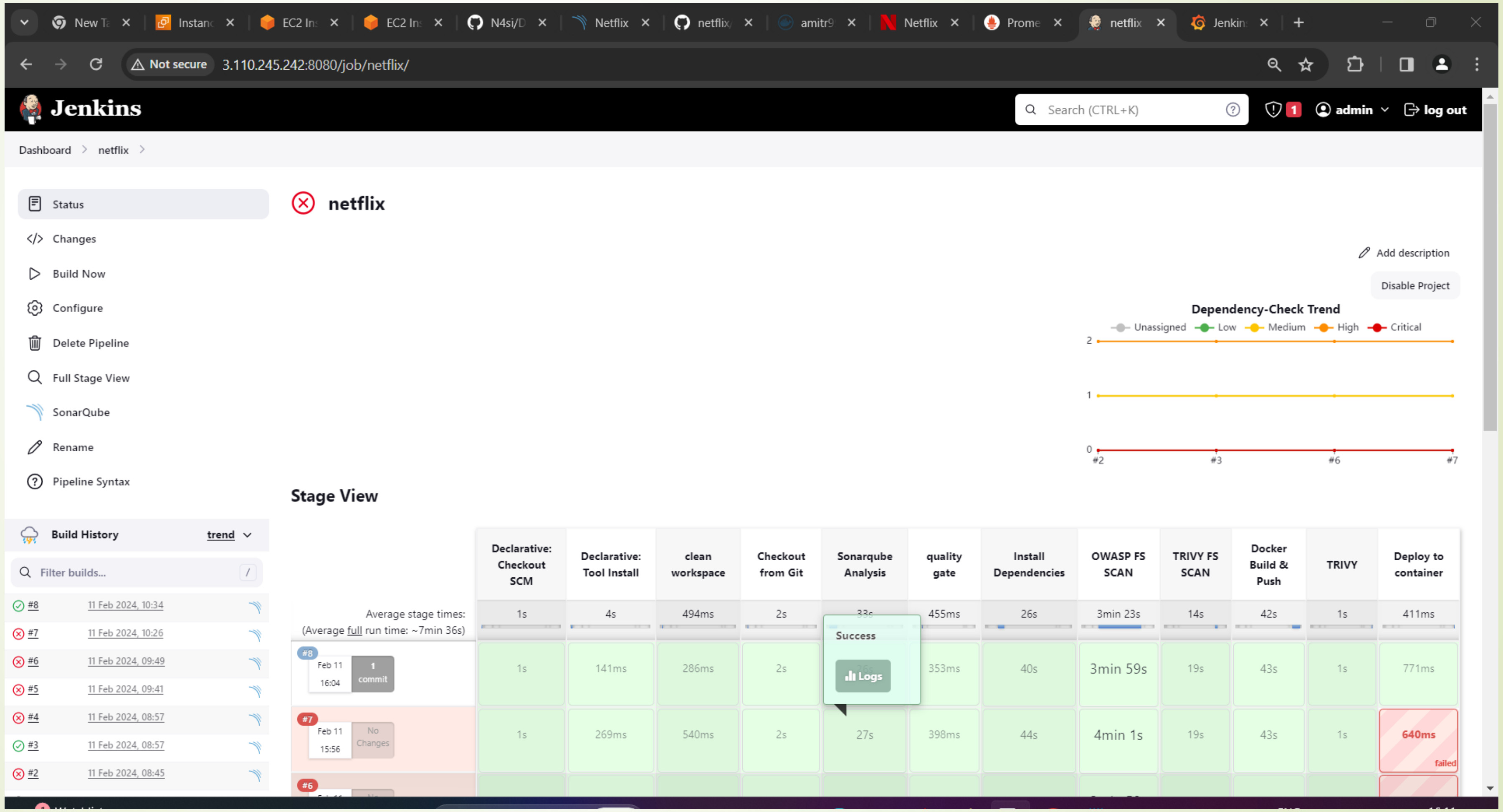
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DN
my_project	i-0db3e7e80d7c91b4d	Running	t2.large	2/2 checks passed	View alarms	ap-south-1b	ec2-3-110-24
monitoring	i-0ca84351a5a29134e	Running	t2.medium	2/2 checks passed	View alarms	ap-south-1b	ec2-13-234-7

Select an instance

© 2024, Amazon Web Services, Inc. or its affiliates.

Project Output (Contd.)

- Jenkins Pipeline



3.110.245.242:8080/job/netflix/

Dashboard > netflix >

netflix

- Status
- Changes
- Build Now
- Configure
- Delete Pipeline
- Full Stage View
- SonarQube
- Rename
- Pipeline Syntax

Build History trend

#8	11 Feb 2024, 10:34
#7	11 Feb 2024, 10:26
#6	11 Feb 2024, 09:49
#5	11 Feb 2024, 09:41
#4	11 Feb 2024, 08:57
#3	11 Feb 2024, 08:57
#2	11 Feb 2024, 08:45

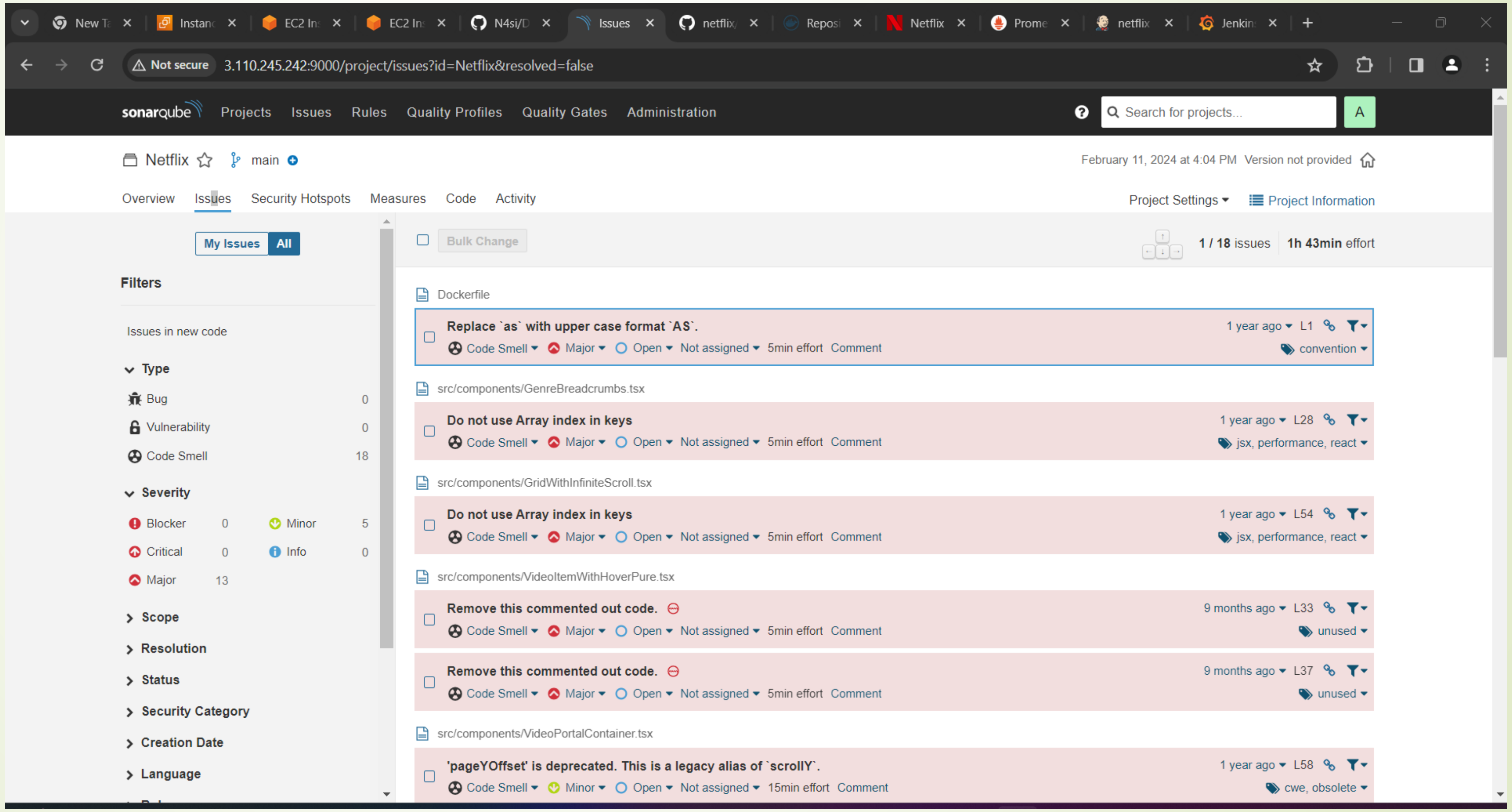
Dependency-Check Trend

Stage View

	Declarative: Checkout SCM	Declarative: Tool Install	clean workspace	Checkout from Git	Sonarqube Analysis	quality gate	Install Dependencies	OWASP FS SCAN	TRIVY FS SCAN	Docker Build & Push	TRIVY	Deploy to container
Average stage times: (Average full run time: ~7min 36s)	1s	4s	494ms	2s	33s	455ms	26s	3min 23s	14s	42s	1s	411ms
#8 Feb 11 16:04 1 commit	1s	141ms	286ms	2s	Success	353ms	40s	3min 59s	19s	43s	1s	771ms
#7 Feb 11 15:56 No Changes	1s	269ms	540ms	2s	27s	398ms	44s	4min 1s	19s	43s	1s	640ms failed

Project Output (Contd.)

- SonarQube Scan Report



New T | Instanc | EC2 In | EC2 In | N4si/D | Issues | netflix | Reposi | Netflix | Prome | netflix | Jenkin

Not secure 3.110.245.242:9000/project/issues?id=Netflix&resolved=false

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

Search for projects...

Netflix main

February 11, 2024 at 4:04 PM Version not provided

Overview Issues Security Hotspots Measures Code Activity

Project Settings Project Information

My Issues All

Filters

Issues in new code

Type

Bug	0
Vulnerability	0
Code Smell	18

Severity

Blocker	0	Minor	5
Critical	0	Info	0
Major	13		

Scope

Resolution

Status

Security Category

Creation Date

Language

Bulk Change

1 / 18 issues 1h 43min effort

Dockerfile

Replace `as` with upper case format `AS`. 1 year ago L1
Code Smell Major Open Not assigned 5min effort Comment convention

src/components/GenreBreadcrumbs.tsx

Do not use Array index in keys 1 year ago L28
Code Smell Major Open Not assigned 5min effort Comment jsx, performance, react

src/components/GridWithInfiniteScroll.tsx

Do not use Array index in keys 1 year ago L54
Code Smell Major Open Not assigned 5min effort Comment jsx, performance, react

src/components/VideoItemWithHoverPure.tsx

Remove this commented out code. 9 months ago L33
Code Smell Major Open Not assigned 5min effort Comment unused

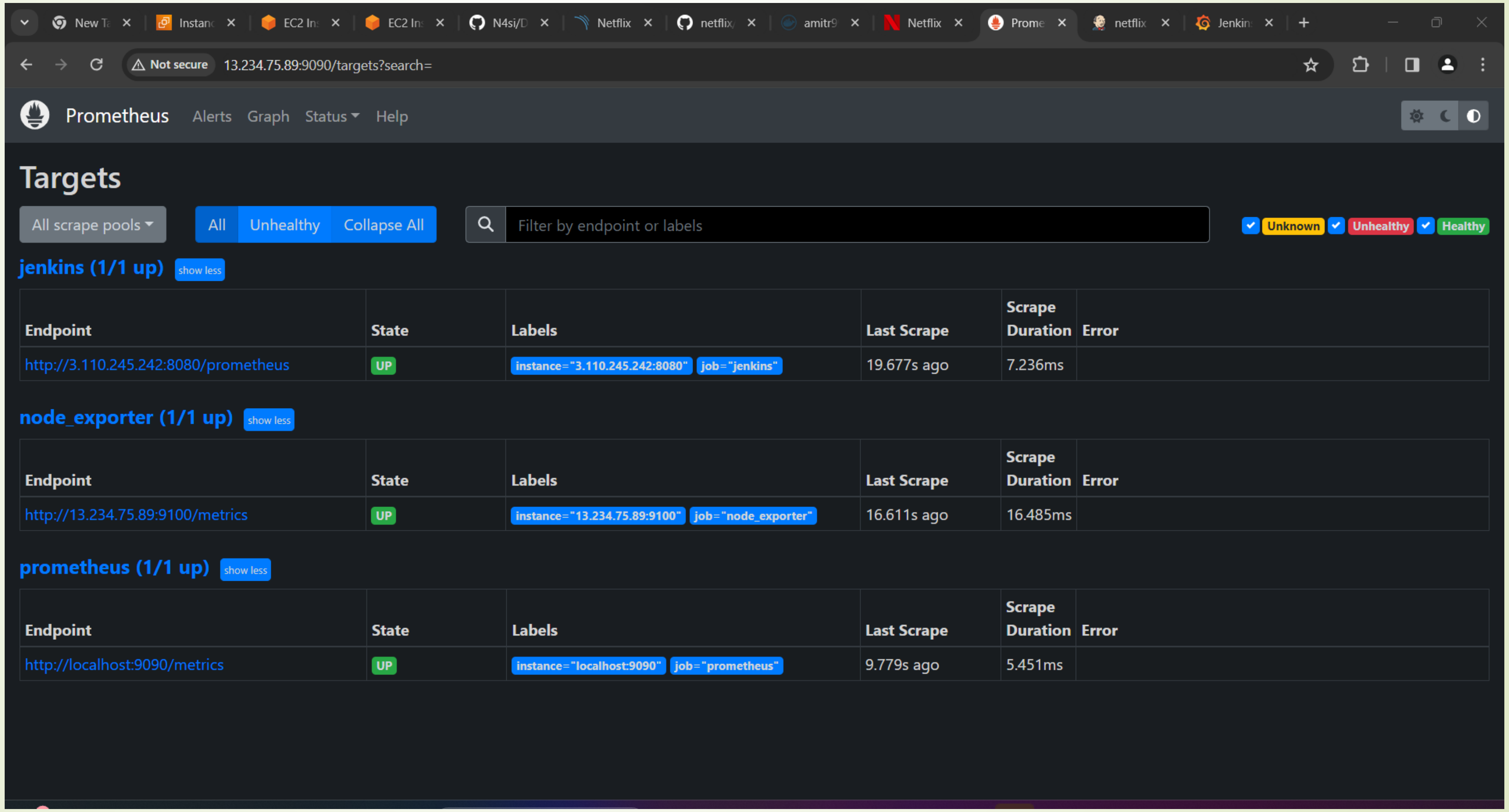
Remove this commented out code. 9 months ago L37
Code Smell Major Open Not assigned 5min effort Comment unused

src/components/VideoPortalContainer.tsx

'pageYOffset' is deprecated. This is a legacy alias of `scrollY`. 1 year ago L58
Code Smell Minor Open Not assigned 15min effort Comment cwe, obsolete

Project Output (Contd.)

- Prometheus



13.234.75.89:9090/targets?search=

Prometheus Alerts Graph Status Help

Targets

All scrape pools | All | Unhealthy | Collapse All | Filter by endpoint or labels | ✓ Unknown ✓ Unhealthy ✓ Healthy

jenkins (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://3.110.245.242:8080/prometheus	UP	instance="3.110.245.242:8080" job="jenkins"	19.677s ago	7.236ms	

node_exporter (1/1 up) show less

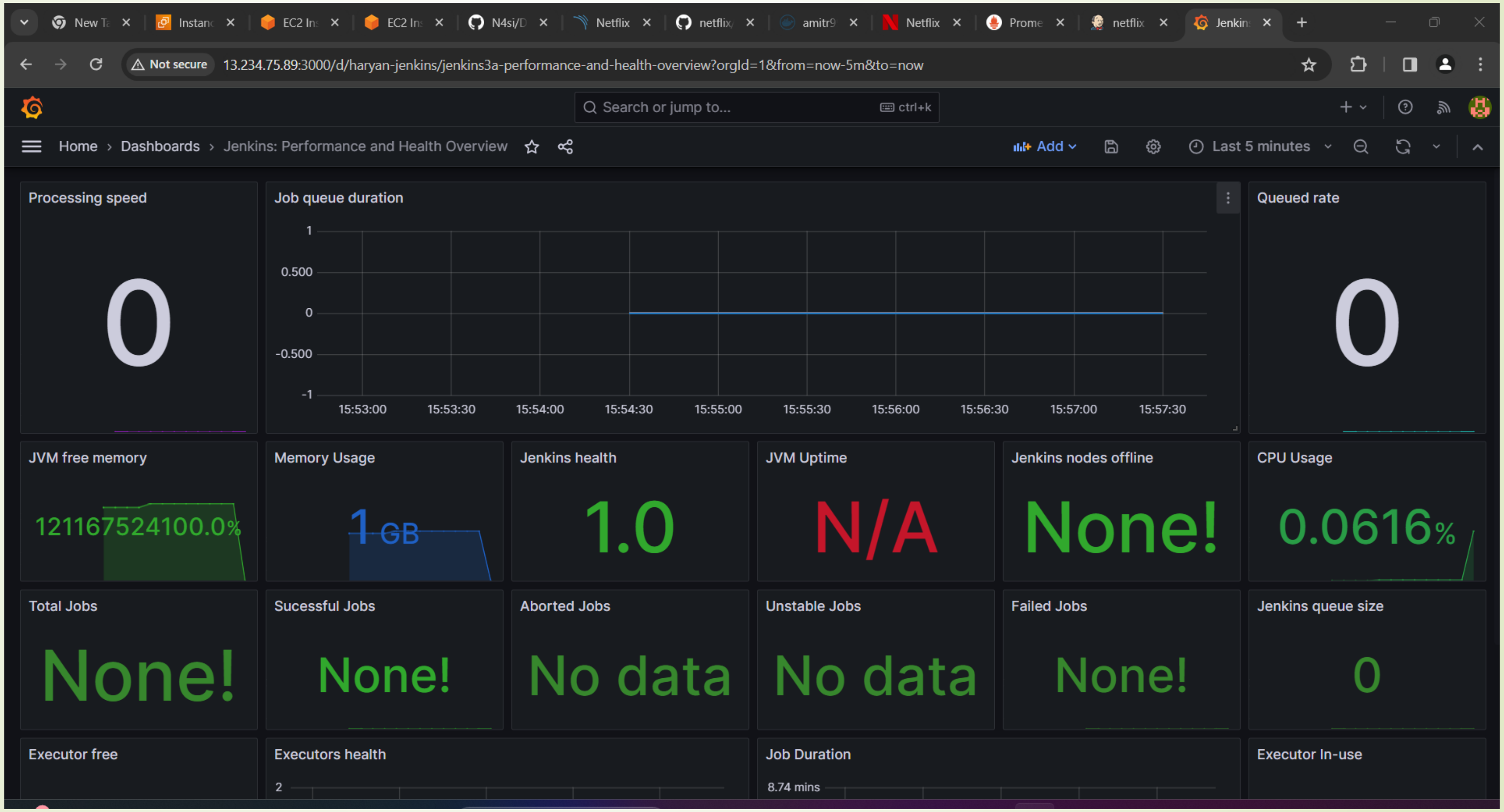
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://13.234.75.89:9100/metrics	UP	instance="13.234.75.89:9100" job="node_exporter"	16.611s ago	16.485ms	

prometheus (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9090/metrics	UP	instance="localhost:9090" job="prometheus"	9.779s ago	5.451ms	

Project Output (Contd.)

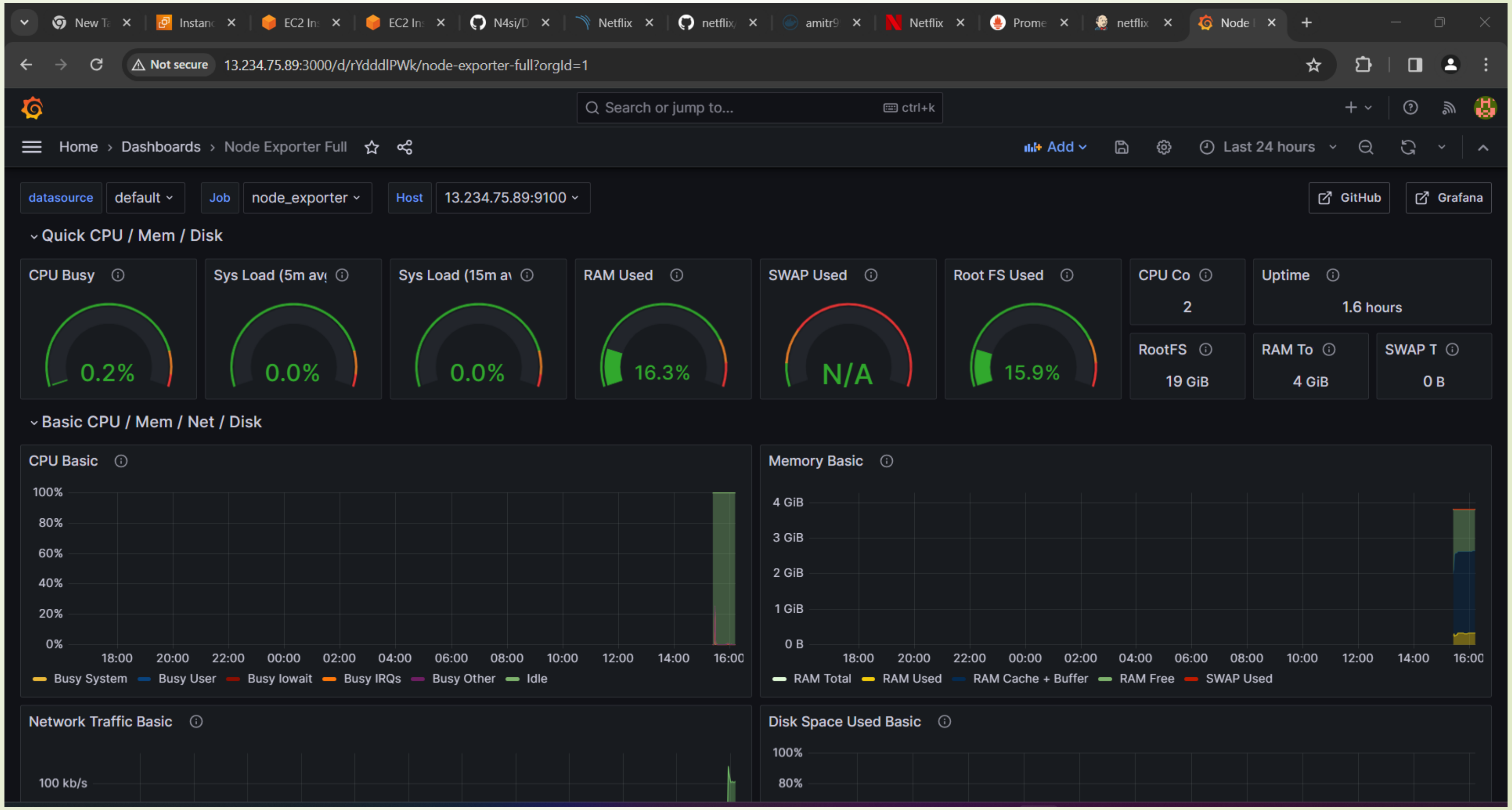
- Grafana



Processing speed: 0
Job queue duration
Queued rate: 0
JVM free memory: 121167524100.0%
Memory Usage: 1 GB
Jenkins health: 1.0
JVM Uptime: N/A
Jenkins nodes offline: None!
CPU Usage: 0.0616%
Total Jobs: None!
Sucessful Jobs: None!
Aborted Jobs: No data
Unstable Jobs: No data
Failed Jobs: None!
Jenkins queue size: 0
Executor free
Executors health
Job Duration
Executor In-use

Project Output (Contd.)

❑ Server Monitoring


13.234.75.89:3000/d/rYdddlPWk/node-exporter-full?orgId=1

Home › Dashboards › Node Exporter Full

datasource: default | Job: node_exporter | Host: 13.234.75.89:9100

Last 24 hours

Quick CPU / Mem / Disk

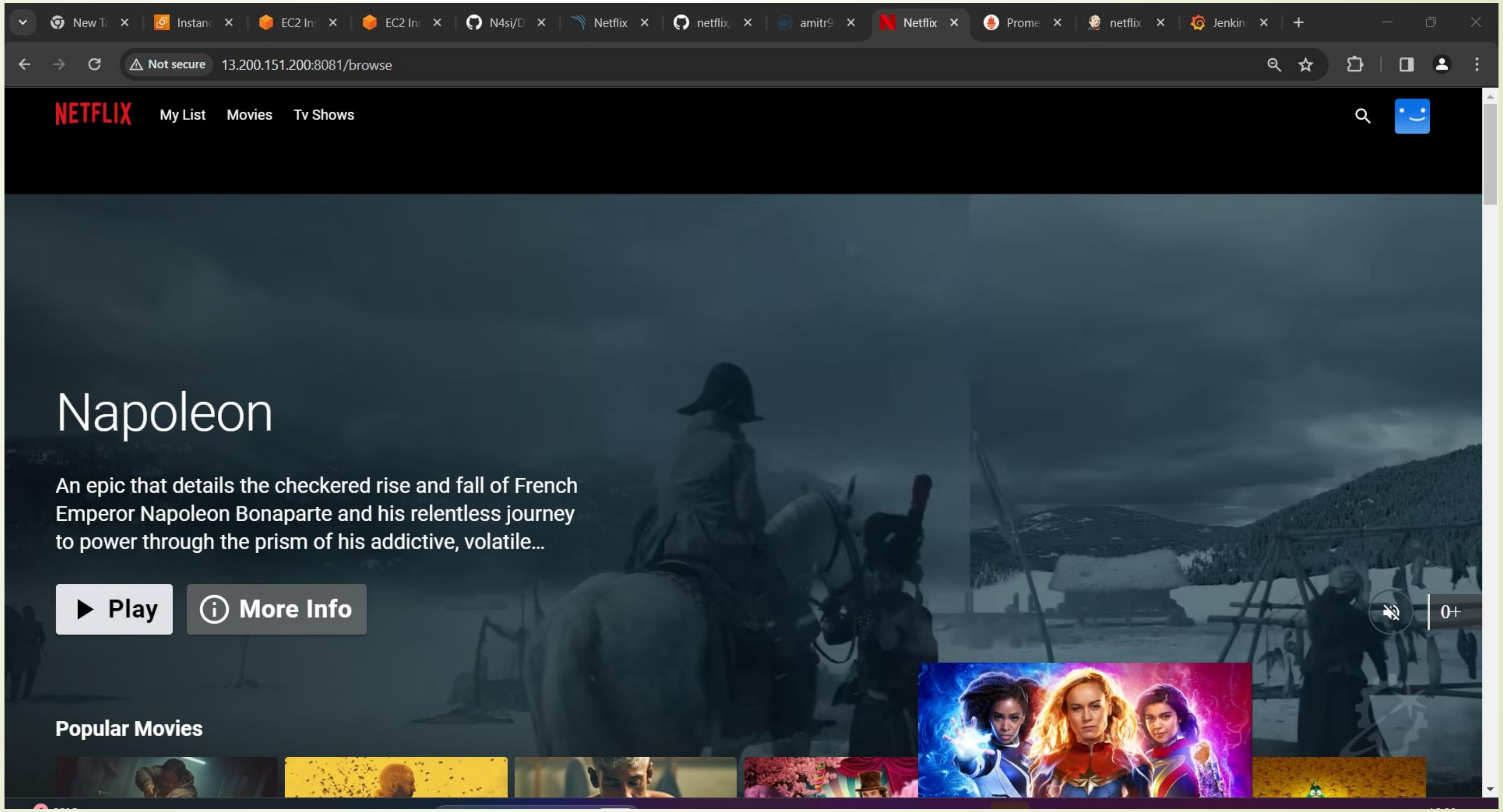
CPU Busy	Sys Load (5m avg)	Sys Load (15m avg)	RAM Used	SWAP Used	Root FS Used	CPU Cores	Uptime
0.2%	0.0%	0.0%	16.3%	N/A	15.9%	2	1.6 hours

RootFS Total	RAM Total	SWAP Total
19 GiB	4 GiB	0 B

Basic CPU / Mem / Net / Disk

Project Output (Contd.)

- Web Application Running



Napoleon

An epic that details the checkered rise and fall of French Emperor Napoleon Bonaparte and his relentless journey to power through the prism of his addictive, volatile...

Popular Movies

Conclusion

- In conclusion, this project offers a complete solution to make it easier to deploy containerized apps while making sure they're very secure right from the start until they're in use. By using technologies such as Jenkins, Docker, Kubernetes, Helm, ArgoCD, and monitoring tools like Prometheus and Grafana, we have demonstrated the ability to automate deployment tasks, ensure scalability, and maintain robust security throughout the software development lifecycle. Through continuous integration, delivery, and security scanning processes, we promoted a culture of collaboration, agility, and security, ultimately leading to the delivery of high-quality, secure software to end-users. As organizations navigate the rapidly evolving technological landscape, adopting DevSecOps practices becomes imperative to stay competitive, reduce risks associated with security breaches, and deliver value to customers efficiently.

Future Scope

- Looking ahead, there are several ways we can improve this project. We could make deployment even easier by adding more automation. Also, we can make it safer by adding better security tools. Exploring new technologies like cloud platforms could help us make everything run smoother and faster. We also want to make sure it works well for different types of programming and is easy for everyone to use. Finally, working with other developers could help us get feedback and make the project even better.