Project Report on

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

Submitted by

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

Under the guidance of

Mr. Sandeep Walvekar

In partial fulfillment of the award of Post Graduate Diploma in

IT Infrastructure, Systems and Security

(PG-DITISS)

Sunbeam Institute of Information Technology,

Pune (Maharashtra)

PG-DITISS -2024

DECLARATION

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included; we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed**.**

Place: Pune

Date:

Rajmane Amit	**Piyush Patil**	**Gaurav Jadhav**	**Raut Kunal**
(230944223030)	**(230944223024)**	**(230944223012)**	**(230944223032)**

CERTIFICATE

This is to certify that the project report entitled **“Deploying Applications on Docker and Kubernetes with Jenkins CI/CD pipeline using AWS and DevSecOps Practices”**, submitted by **Rajmane Amit** is the bonafide work completed under our supervision and guidance in partial fulfillment for the award of Post Graduate Diploma in IT Infrastructure, Systems and Security (PG-DITISS) of Sunbeam Institute of Information Technology, Pune (M.S.).

Place: Pune

Date:

Mr. Sandeep Walvekar	**Mr. Vishal Salunkhe**
Guide	Course Coordinator

Mr. Nitin Kudale

CEO

Sunbeam Institute of Information Technology

Pune (M.S.) – 411057

CERTIFICATE

This is to certify that the project report entitled **“Deploying Applications on Docker and Kubernetes with Jenkins CI/CD pipeline using AWS and DevSecOps Practices”**, submitted by **Piyush Patil** is the bonafide work completed under our supervision and guidance in partial fulfillment for the award of Post Graduate Diploma in IT Infrastructure, Systems and Security (PG-DITISS) of Sunbeam Institute of Information Technology, Pune (M.S.).

Place: Pune

Date:

Mr. Sandeep Walvekar	**Mr. Vishal Salunkhe**
Guide	Course Coordinator

Mr. Nitin Kudale

CEO

Sunbeam Institute of Information Technology

Pune (M.S.) – 411057

CERTIFICATE

This is to certify that the project report entitled **“Deploying Applications on Docker and Kubernetes with Jenkins CI/CD pipeline using AWS and DevSecOps Practices”**, submitted by **Gaurav Jadhav** is the bonafide work completed under our supervision and guidance in partial fulfillment for the award of Post Graduate Diploma in IT Infrastructure, Systems and Security (PG-DITISS) of Sunbeam Institute of Information Technology, Pune (M.S.).

Place: Pune

Date:

Mr. Sandeep Walvekar	**Mr. Vishal Salunkhe**
Guide	Course Coordinator

Mr. Nitin Kudale

CEO

Sunbeam Institute of Information Technology

Pune (M.S.) – 411057

CERTIFICATE

This is to certify that the project report entitled **“Deploying Applications on Docker and Kubernetes with Jenkins CI/CD pipeline using AWS and DevSecOps Practices”**, submitted by **Raut Kunal** is the bonafide work completed under our supervision and guidance in partial fulfillment for the award of Post Graduate Diploma in IT Infrastructure, Systems and Security (PG-DITISS) of Sunbeam Institute of Information Technology, Pune (M.S.).

Place: Pune

Date:

Mr. Sandeep Walvekar	**Mr. Vishal Salunkhe**
Guide	Course Coordinator

Mr. Nitin Kudale

CEO

Sunbeam Institute of Information Technology

Pune (M.S.) – 411057

APPROVAL CERTIFICATE

This Project II report entitled **“Deploying Applications on Docker and Kubernetes with Jenkins CI/CD pipeline using AWS and DevSecOps Practices”** by **Rajmane Amit (230944223030)** is approved for Post Graduate Diploma in IT Infrastructure, Systems and Security (PG-DITISS) of Sunbeam Institute of Information Technology, Pune (M.S.).

Place: Pune

Date:

Examiner: ________________________

(Signature)

____

(Name)

APPROVAL CERTIFICATE

This Project II report entitled **“Deploying Applications on Docker and Kubernetes with Jenkins CI/CD pipeline using AWS and DevSecOps Practices”** by **Piyush Patil (230944223024)** is approved for Post Graduate Diploma in IT Infrastructure, Systems and Security (PG-DITISS) of Sunbeam Institute of Information Technology, Pune (M.S.).

Place: Pune

Date:

Examiner: ______________________

(Signature)

____

(Name)

APPROVAL CERTIFICATE

This Project II report entitled **“Deploying Applications on Docker and Kubernetes with Jenkins CI/CD pipeline using AWS and DevSecOps Practices”** by **Gaurav Jadhav (230944223012)** is approved for Post Graduate Diploma in IT Infrastructure, Systems and Security (PG-DITISS) of Sunbeam Institute of Information Technology, Pune (M.S.).

Place: Pune

Date:

Examiner: ______________________

(Signature)

____

(Name)

APPROVAL CERTIFICATE

This Project II report entitled **“Deploying Applications on Docker and Kubernetes with Jenkins CI/CD pipeline using AWS and DevSecOps Practices”** by **Raut Kunal (230944223032)** is approved for Post Graduate Diploma in IT Infrastructure, Systems and Security (PG-DITISS) of Sunbeam Institute of Information Technology, Pune (M.S.).

Place: Pune

Date:

Examiner: ____________________

(Signature)

____

(Name)

CONTENTS

ABSTRACT

In today's fast-paced world of software, getting applications up and running quickly and safely is crucial. This project focuses on making that happen by using Docker, Kubernetes, Jenkins CI/CD, and DevSecOps practices to simplify the process while keeping everything secure.

We use Kubernetes to organize and manage our applications in containers, making them easier to handle and ensuring they run smoothly no matter where they're deployed.

Jenkins CI/CD Pipeline helps us automate the steps of building, testing, and delivering our apps. This means less manual work and faster results, so we can get our apps out to users more efficiently.

To keep our apps secure, we've added extra tools and practices. Helm helps us manage our Kubernetes apps consistently, while Prometheus and Grafana keep an eye on how our apps are performing.

We also use SonarQube to check our code for any potential issues early on, and tools like Trivy and OWASP to scan our container images for vulnerabilities and make sure they meet security standards.

By combining these tools and practices, we're able to automate many parts of the deployment process, making it faster, more reliable, and safer. This helps us work together better as a team and deliver better-quality software to our users faster.

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. To address these challenges, 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. Furthermore, integrating DevSecOps practices into the deployment process has become imperative to ensure robust security measures throughout the software development lifecycle.

This project aims to bridge the gap between agility, scalability, and security by streamlining the deployment process of containerized applications while embedding stringent security measures from inception to production. Leveraging Kubernetes as the container orchestration platform provides the foundation for automating deployment tasks, managing application scalability, and ensuring resilience across diverse environments. Kubernetes' declarative configuration and self-healing capabilities empower teams to focus more on application logic and less on infrastructure management, thereby accelerating the deployment cycle.

The integration of Jenkins CI/CD Pipeline into the deployment workflow further enhances agility and reliability by automating the building, testing, and delivery of containerized applications. With Jenkins, teams can establish automated pipelines that facilitate rapid feedback loops, enabling developers to iterate quickly and deploy changes efficiently. By automating mundane tasks and standardizing deployment processes, Jenkins promotes consistency, reduces human error, and accelerates time-to-market.

However, agility and speed cannot come at the expense of security. To fortify the deployment pipeline, this project incorporates various DevSecOps tools and practices, including Helm, Prometheus, Grafana, SonarQube, Trivy, and OWASP. Helm simplifies the management of Kubernetes applications by providing templating and versioning capabilities, ensuring consistency and repeatability in deployments. Prometheus and Grafana offer robust monitoring and visualization features, allowing teams to monitor application performance and identify potential issues proactively.

Moreover, SonarQube plays a pivotal role in maintaining code quality by conducting static code analysis, identifying bugs, vulnerabilities, and code smells early in the development cycle. This enables developers to address issues promptly, reducing technical debt and enhancing overall software quality. Additionally, Trivy and OWASP contribute to container security by scanning images for vulnerabilities and enforcing best security practices, thereby mitigating risks associated with security breaches and compliance violations.

By integrating these DevSecOps tools into the deployment pipeline, the project aims to automate deployment, continuous integration, delivery, and security scanning processes, fostering a culture of collaboration, agility, and security across the software development lifecycle. This not only accelerates the deployment of containerized applications but also ensures that security remains a top priority throughout the development process, ultimately leading to the delivery of high-quality, secure software to end-users.

1.1 Benefits

- Streamlined Deployments: Manual interventions are minimized, leading to faster releases and shorter feedback loops.
- Continuous Delivery: New features and bug fixes are delivered frequently, enhancing user experience and reducing time-to-market.
- Enhanced Security: By integrating security throughout the pipeline, vulnerabilities are identified and addressed early, minimizing potential security breaches.

- Scalability and High Availability: Kubernetes ensures applications can seamlessly scale up or down based on demand, while self-healing capabilities keep them running even in case of failures.
- Cloud-Based Infrastructure: AWS provides a pay-as-you-go model, offering flexibility and cost-efficiency.

1.2 Project Plan

Table: Activities Details

Sr. No.	**ACTIVITY**	**WEEK**			
		1	**2**	**3**	**4**
1	Project group formation	🟨			
2	Project work to be started in respective labs	🟨			
3	First review with PPT presentation		🟨		
4	Design Use-Case view as per project			🟨	
5	Design Block diagram as per project			🟨	
6	Second review with PPT presentation				🟨
7	Selection			🟨	
8	Final review with PPT presentation			🟨	
9	Implementation coding as per project			🟨	
10	Testing, Troubleshooting with different techniques			🟨	🟨
11	Created Soft copy of project and then final hard copy				🟨

2. LITERATURE SURVEY

Paper 1: Container orchestration and its implications on enterprise networking

Author: Mohanty, B., & Sarma, S.

Description: This paper explores the implications of container orchestration on enterprise networking. It discusses the challenges and benefits of using container orchestration technologies like Kubernetes for managing containerized applications in enterprise environments. The authors delve into topics such as scalability, resilience, and automation provided by Kubernetes, highlighting its impact on enterprise networking practices. The paper provides valuable insights into the role of container orchestration in modernizing enterprise infrastructure and streamlining application deployment processes.

Paper 2: Continuous Integration and Continuous Delivery in the IT Industry: A Case Study

Author: Stensholt, E., Landmark, A., & Fard, S. F.

Description: This paper presents a case study focusing on the implementation of Continuous Integration (CI) and Continuous Delivery (CD) practices in the IT industry. The authors investigate the adoption and impact of CI/CD pipelines on software development processes, highlighting their role in improving productivity, code quality, and release frequency. Through empirical analysis and insights from industry practitioners, the paper discusses challenges, benefits, and lessons learned from CI/CD implementation. It provides valuable recommendations for organizations seeking to streamline their software delivery pipelines and embrace CI/CD methodologies effectively.

Paper 3: DevSecOps: A Software Development Process for Securing Cloud-Native Applications

Author: Chapman, C., Muthanna, A., & Rajani, S.

Description: This paper introduces the concept of DevSecOps, emphasizing the integration of security practices into the software development lifecycle for cloud-native applications. The authors discuss the principles, benefits, and challenges of implementing DevSecOps methodologies, focusing on ensuring security without compromising agility. Through case studies and practical examples, the paper demonstrates how DevSecOps practices can enhance application security, mitigate risks, and foster collaboration among development, operations, and security teams. It serves as a valuable resource for organizations transitioning to cloud-native environments and seeking to adopt a holistic approach to application security.

3. SYSTEM DEVELOPMENT AND DESIGN

3.1 Proposed System

The system architecture comprises two Amazon EC2 instances: a t2.large instance hosting Jenkins, OWASP Scan, Docker, Trivy, and Amazon Ubuntu AMI for the base operating system, and a t2.medium instance dedicated to hosting Prometheus and Grafana for monitoring purposes. Additionally, Amazon Elastic Kubernetes Service (EKS) is utilized to orchestrate and manage containerized applications, with Helm for package management and ArgoCD for GitOps-based continuous delivery.

Jenkins serves as the CI/CD server, automating the build, test, and deployment processes of applications. It integrates seamlessly with Docker and EKS for managing containerized deployments. OWASP Scan conducts security scans on applications deployed through Jenkins to identify vulnerabilities, while Trivy scans container images pre-deployment to ensure secure deployments. Helm streamlines application deployment and management on Kubernetes, while ArgoCD automates deployment based on Git repository changes, ensuring consistency and reliability.

The integration between components is seamless and efficient. Jenkins orchestrates the build, test, and deployment processes, utilizing Docker for containerization and EKS for deployment orchestration. Security scanning tools OWASP Scan and Trivy integrate with Jenkins to ensure secure deployments through pre-deployment and post-deployment vulnerability assessments. Helm and ArgoCD automate the deployment process, ensuring consistency and reliability through GitOps practices. Monitoring tools Prometheus and Grafana collect and visualize metrics from EKS and other components for continuous monitoring and analysis.

The system is designed to be scalable and highly available. Amazon EKS automatically scales based on workload demands, ensuring optimal resource utilization.

3.2 System Architecture

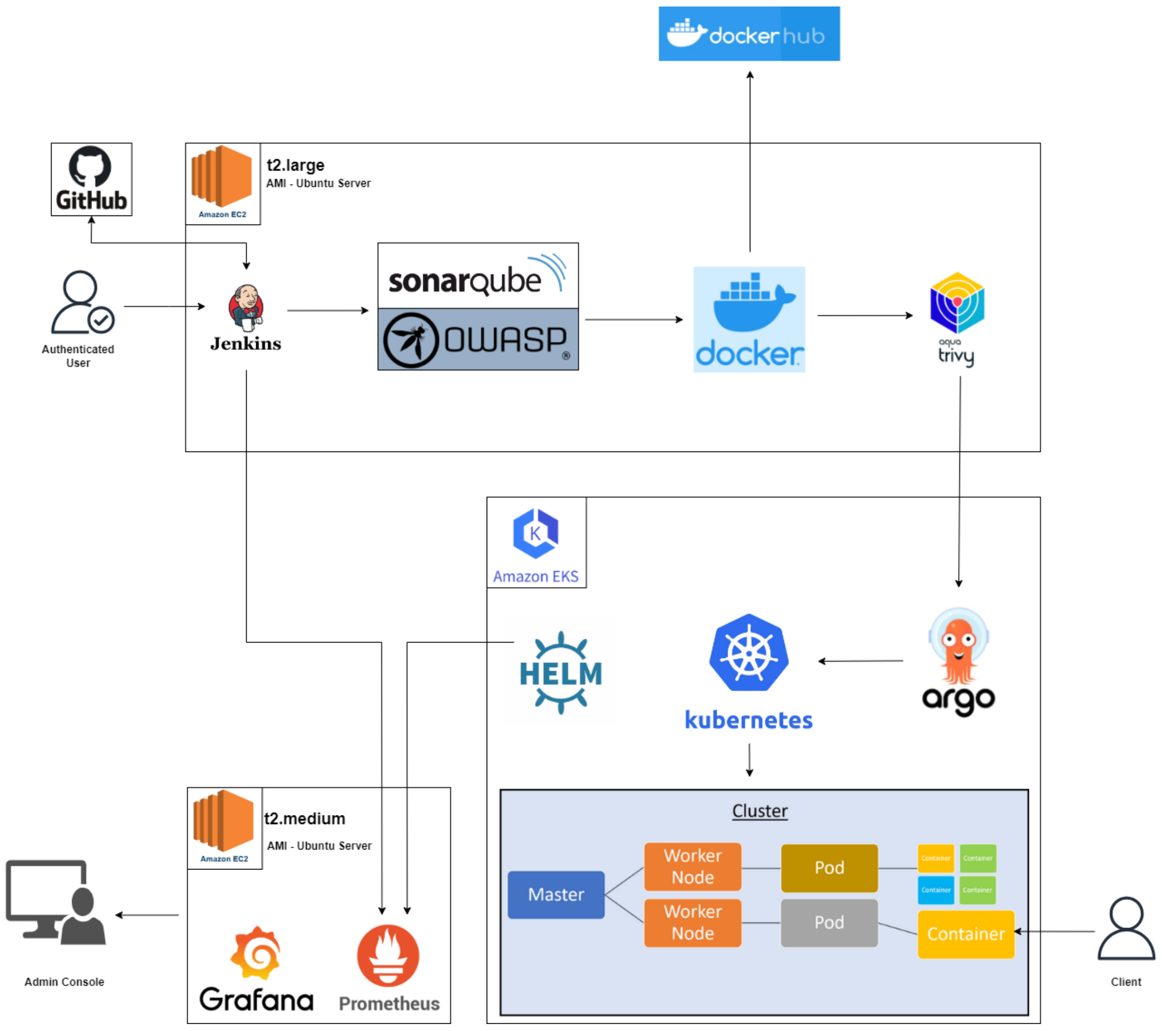


Figure: System Architecture

3.3 Technology used

3.3.1 Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) is a web service provided by Amazon it allows you to rent virtual servers in the cloud, known as instances, to run your applications and workloads. EC2 provides a scalable and flexible infrastructure that enables you to quickly deploy and manage virtual servers without the need to invest in physical hardware.

Key features

Instances: These are virtual servers that you can launch and manage in the cloud. You can choose from a wide range of instance types optimized for different use cases, such as general-purpose computing, memory-intensive tasks, high-performance computing, and more.

Elasticity and Scaling: EC2 allows you to easily scale your infrastructure up or down based on your workload's demands. You can create multiple instances or change the instance type to handle varying levels of traffic.

Amazon Machine Images (AMIs): AMIs are pre-configured templates that contain an operating system and often additional software needed to launch an instance. You can use AWS-provided AMIs or create your own custom AMIs.

Security Groups: Security groups act as virtual firewalls that control inbound and outbound traffic to your instances. You can define rules to allow or deny specific types of traffic.

Cost Management: EC2 offers various pricing options, including On-Demand Instances, Reserved Instances, and Spot Instances, which provide flexibility in managing costs based on your usage patterns.

3.3.2 Git

Git is a distributed version control system (VCS) designed to manage source code history and facilitate collaborative software development.

Key features of Git:

Distributed Architecture: Unlike centralized version control systems, Git is distributed. Each developer has a complete copy of the repository, including its entire history. This allows for offline work, faster operations, and improved resilience.

Branching and Merging: Git makes it easy to create branches, which are separate lines of development. Developers can work on features, bug fixes, or experiments in their own branches without affecting the main codebase. Merging branches back together is relatively simple and allows for collaborative development.

Commit History: Git maintains a detailed history of changes to the codebase. Each change is represented by a commit, which includes information about who made the change, when it was made, and what was changed. This commit history provides a clear view of the evolution of the project.

Fast and Efficient: Git is designed for speed and efficiency. Most operations are local, as the repository resides on the developer's machine. This results in rapid commits, branching, and merging.

Collaboration: Git enables effective collaboration among developers. Multiple developers can work on different branches simultaneously, and changes can be shared by pushing them to a remote repository. Pull requests or merge requests facilitate the process of reviewing and integrating changes from different contributors.

3.3.3 Docker

Docker is an open-source platform that allows you to automate the deployment, scaling, and management of applications using containerization. Containers are lightweight, portable, and isolated environments that package an application and its dependencies together.

Key features of Docker:

Containerization: Docker allows you to create containers that encapsulate applications and their dependencies, including libraries, runtime, and system tools. Containers ensure consistency between different environments, from development laptops to production servers.

Portability: Docker containers can run consistently across different environments, such as local development machines, testing servers, and cloud-based production environments. This eliminates the "it works on my machine" problem.

Efficiency: Containers share the host operating system's kernel and resources, which makes them more lightweight and efficient compared to traditional virtual machines. This leads to faster startup times and reduced resource consumption.

Version Control: Docker images are versioned, allowing you to track changes to your application and its dependencies over time. This facilitates rollback and ensures that you can recreate previous versions of your application easily.

Docker Images: Docker images are the blueprints for containers. They are created from a set of instructions defined in a Dockerfile, which specifies the base image, environment, configuration, and application code.

3.3.4 Jenkins

Jenkins is an open-source automation server that facilitates the continuous integration and continuous delivery (CI/CD) of software projects. It helps automate various tasks related to building, testing, and deploying applications, making the development and release process more efficient and reliable.

Key features of Jenkins:

Continuous Integration: Jenkins automates the process of integrating code changes from multiple contributors into a shared repository. It triggers builds whenever code is committed, allowing developers to identify and fix integration issues early.

Automated Builds: Jenkins can automatically build projects from source code repositories. It supports various build tools, languages, and platforms, making it versatile for different types of projects.

Extensibility: Jenkins can be extended through a wide range of plugins that provide additional functionalities. Plugins are available for source code management, build tools, testing frameworks, and deployment options.

Pipeline as Code: Jenkins uses a domain-specific language called Groovy to define build pipelines as code. This enables you to define complex workflows that include build, test, and deployment stages in a version-controlled script.

Continuous Delivery: Jenkins supports continuous delivery by automating the deployment process after successful builds. It can deploy applications to different environments, such as development, staging, and production.

Distributed Builds: Jenkins can distribute builds across multiple machines, allowing for parallel builds and improved build performance. This is particularly useful for large and resource-intensive projects.

3.3.5 Amazon EKS

Amazon EKS (Elastic Kubernetes Service) is a managed Kubernetes service offered by Amazon Web Services (AWS) that simplifies the deployment, management, and scaling of containerized applications using Kubernetes.

Key features of Amazon EKS include:

Managed Kubernetes Control Plane: EKS manages the Kubernetes control plane, including the control plane nodes, API server, etcd, and scheduler. This allows users to focus on deploying and managing their applications without worrying about the underlying infrastructure.

High Availability and Scalability: EKS provides high availability by distributing control plane nodes across multiple Availability Zones (AZs) within a region. This ensures redundancy and fault tolerance, minimizing the risk of downtime. EKS automatically scales to meet the demands of containerized applications, allowing clusters to scale horizontally by adding or removing worker nodes based on workload requirements.

Cost Optimization: EKS offers cost optimization features, such as Reserved Instances (RIs) and Savings Plans, which allow users to save costs by committing to a certain level of usage over a specified period. EKS also provides cost visibility and management through detailed billing and cost allocation reports, allowing users to monitor and optimize their Kubernetes infrastructure costs.

3.3.6 SonarQube

SonarQube is an open-source platform developed by SonarSource for continuous inspection of code quality to perform automatic reviews with static analysis of code to detect bugs, code smells, and security vulnerabilities on 20+ programming languages. It provides a centralized dashboard to view metrics on code quality, security vulnerabilities, and overall project health.

Key features of SonarQube include:

Static Code Analysis: SonarQube analyzes source code without executing it to identify bugs, code smells, security vulnerabilities, and potential performance issues.

Code Quality Metrics: It provides a range of metrics such as code coverage, duplication, complexity, and maintainability, helping developers understand and improve the overall quality of their codebase.

Security Analysis: SonarQube includes security-specific rules to identify vulnerabilities such as SQL injection, Cross-Site Scripting (XSS), and other security weaknesses in the codebase.

Code Duplication Detection: SonarQube identifies duplicated code blocks within the project, helping developers refactor and maintain a cleaner codebase.

3.3.7 OWASP

OWASP (Open Web Application Security Project) provides a variety of resources and tools for improving the security of web applications. An OWASP scan typically refers to the process of using tools and methodologies provided by OWASP to assess the security posture of web applications, identify potential vulnerabilities, and address them to enhance overall security. One of the key tools provided by OWASP for this purpose is OWASP ZAP (Zed Attack Proxy).

Key features of OWASP ZAP include:

Automated Scanning: OWASP tools like ZAP provide automated scanning capabilities to identify common security vulnerabilities in web applications, such as SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF), insecure configurations, and more.

Active and Passive Scanning: OWASP ZAP allows for both active and passive scanning of web applications. Active scanning involves sending requests to the target application to identify vulnerabilities, while passive scanning involves monitoring traffic and identifying vulnerabilities based on observed patterns and behaviors.

Reporting: OWASP ZAP generates comprehensive reports detailing the findings of security scans, including identified vulnerabilities, their severity levels, and recommendations for remediation. These reports are invaluable for communicating security risks to development teams and stakeholders.

3.3.8 Trivy

Trivy is an open-source vulnerability scanner for containers and other artifacts. It is primarily used to scan container images to identify vulnerabilities within them. Trivy is designed to be fast, simple, and easy to integrate into containerized environments.

Key features of Trivy include:

Vulnerability Detection: Trivy scans container images and other artifacts to detect known vulnerabilities present in their dependencies, including operating system packages, libraries, and application dependencies.

Fast Scanning: Trivy is optimized for speed, enabling fast vulnerability scanning of container images. This makes it suitable for use in CI/CD pipelines and other automated workflows where quick feedback is essential.

Support for CVEs and CVSS Scores: Trivy provides information about Common Vulnerabilities and Exposures (CVEs) associated with detected vulnerabilities, along with their corresponding Common Vulnerability Scoring System (CVSS) scores to assess their severity.

3.3.9 ArgoCD

ArgoCD is an open-source continuous delivery tool for Kubernetes applications. It provides a declarative approach to managing and automating the deployment of applications to Kubernetes clusters. ArgoCD uses GitOps principles, where the desired state of applications and their configurations are defined in Git repositories, and ArgoCD continuously monitors these repositories to ensure that the actual state of the applications matches the desired state.

Key features of ArgoCD include:

Declarative Configuration Management: With ArgoCD, application configurations are managed declaratively using Kubernetes manifests and other YAML files. Developers define the desired state of applications in Git repositories, and ArgoCD ensures that Kubernetes clusters are configured accordingly, simplifying the management of complex application deployments.

Automated Synchronization: ArgoCD continuously monitors Git repositories for changes to application configurations. When changes are detected, ArgoCD automatically synchronizes the configuration changes with the target Kubernetes clusters, ensuring that applications are deployed and updated according to the latest configurations defined in Git.

Integration with CI/CD Pipelines: ArgoCD integrates seamlessly with CI/CD pipelines, enabling automated application deployments as part of the continuous delivery process. Developers can trigger deployments directly from CI/CD pipelines or other automation tools, ensuring that applications are deployed consistently and reliably across different environments.

Customization and Extensibility: ArgoCD is highly customizable and extensible, allowing users to define custom workflows, plugins, and extensions to meet specific requirements. Customization options include defining custom validation rules, pre- and post-sync hooks, and integrating with external systems and tools.

3.3.10 Helm

Helm is a package manager for Kubernetes applications. It simplifies the process of deploying, managing, and updating applications on Kubernetes clusters by providing a way to define, install, and upgrade applications using Helm charts. Helm charts are packages of pre-configured Kubernetes resources, such as deployments, services, and config maps, which can be easily shared and reused.

Key features of Helm include:

Packaging: Helm allows you to package Kubernetes manifests into reusable, versioned packages called Helm charts. These charts encapsulate all the necessary Kubernetes resources and configurations needed to deploy an application.

Security: Helm provides mechanisms for signing and verifying charts to ensure their authenticity and integrity. This helps in maintaining the security of deployments by preventing the installation of tampered or malicious charts.

Release Management: Helm manages releases of applications on Kubernetes clusters. Each deployment of a Helm chart is considered a release, and Helm tracks the state of these releases, making it easy to upgrade, rollback, or uninstall applications.

3.3.11 Prometheus

Prometheus is an open-source monitoring and alerting toolkit originally built at SoundCloud. It is now a standalone open-source project maintained by the Cloud Native Computing Foundation (CNCF). Prometheus is designed for monitoring highly dynamic containerized environments like those orchestrated by Kubernetes.

Key features of Prometheus include:

PromQL: Prometheus Query Language (PromQL) is a powerful query language used to retrieve and manipulate metrics stored in Prometheus. PromQL supports a wide range of operations including filtering, aggregation, and mathematical operations, allowing users to perform complex analyses on their monitoring data.

Histograms and Summaries: Prometheus supports the collection of summarized data such as histograms and summaries, in addition to raw time-series data. This allows users to monitor latency distributions and other aggregated metrics.

Scalability: Prometheus is designed to be highly scalable, both in terms of the number of metrics it can handle and the number of instances it can run. It uses a pull-based model where Prometheus servers scrape metrics from monitored targets at regular intervals. This allows Prometheus to scale horizontally by adding more Prometheus servers as the monitoring load increases.

3.3.12 Grafana

Grafana is an open-source analytics and visualization platform primarily focused on time-series data. It allows users to query, visualize, and understand metrics from various data sources, including databases, cloud monitoring services, and custom applications. Grafana is commonly used for monitoring, observability, and data exploration in a wide range of domains, including DevOps, IoT, and business intelligence.

Key features of Grafana include:

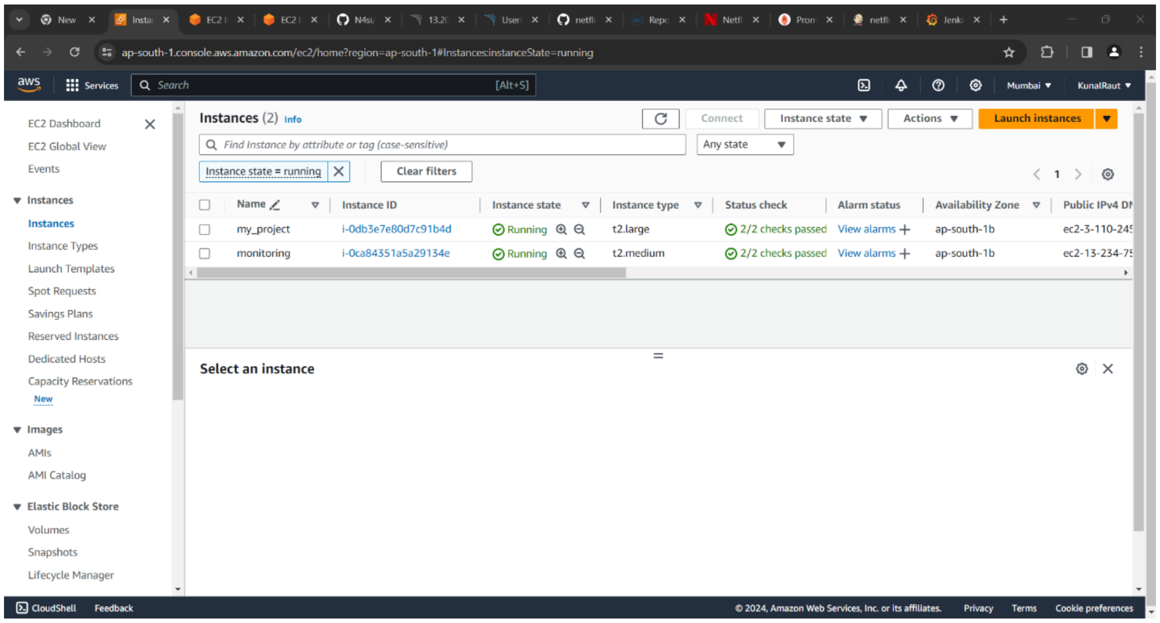
Rich Visualization Options: Grafana provides a wide range of visualization options, including graphs, charts, tables, heatmaps, and gauges. Users can customize the appearance of visualizations using features such as color themes, annotations, and thresholds.

Dashboarding: Grafana allows users to create dynamic dashboards by combining multiple panels and visualizations into a single layout. Dashboards can be customized with interactive features like variable substitution, templating, and drill-down navigation.

Plugins and Extensions: Grafana has a vibrant plugin ecosystem, with thousands of community-contributed plugins and extensions available for extending its functionality. Plugins can add support for new data sources, visualization types, authentication methods, and more.

4. Project Output

4.1 AWS EC2 and EKS


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

Instances (2) Info

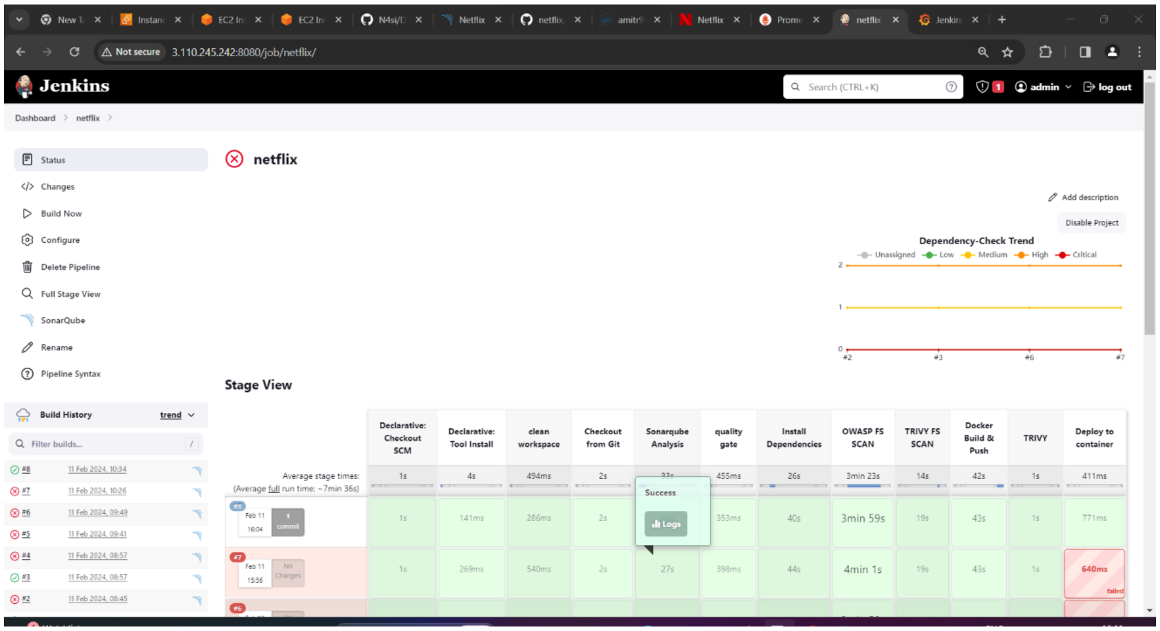
Instance state = running

	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	ec2-3-110-245
☐	monitoring	i-0ca84351a5a29134e	Running	t2.medium	2/2 checks passed	View alarms +	ap-south-1b	ec2-13-234-75

Select an instance

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

4.2 Jenkins Pipeline


3.110.245.242:8080/job/netflix/

Dashboard > netflix >

netflix

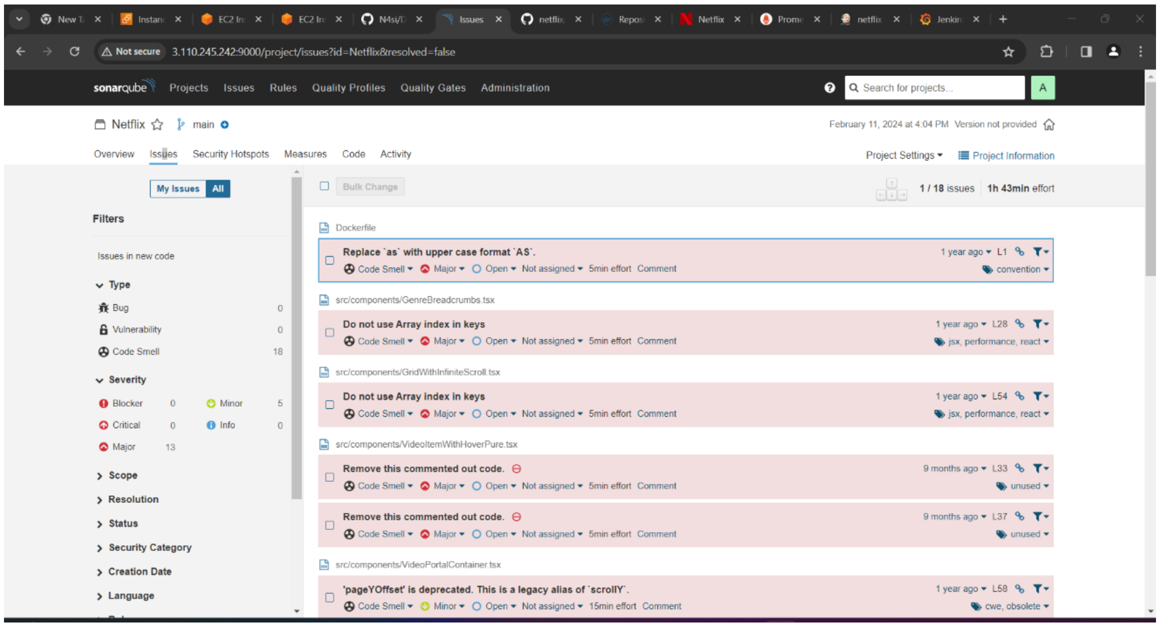
Build History

✓	#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

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

4.3 SonarQube



3.110.245.242:9000/project/issues?id=Netflix&resolved=false

Netflix · main

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

1 / 18 issues 1h 43min effort

Type	
Bug	0
Vulnerability	0
Code Smell	18

Severity			
Blocker	0	Minor	5
Critical	0	Info	0
Major	13		

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

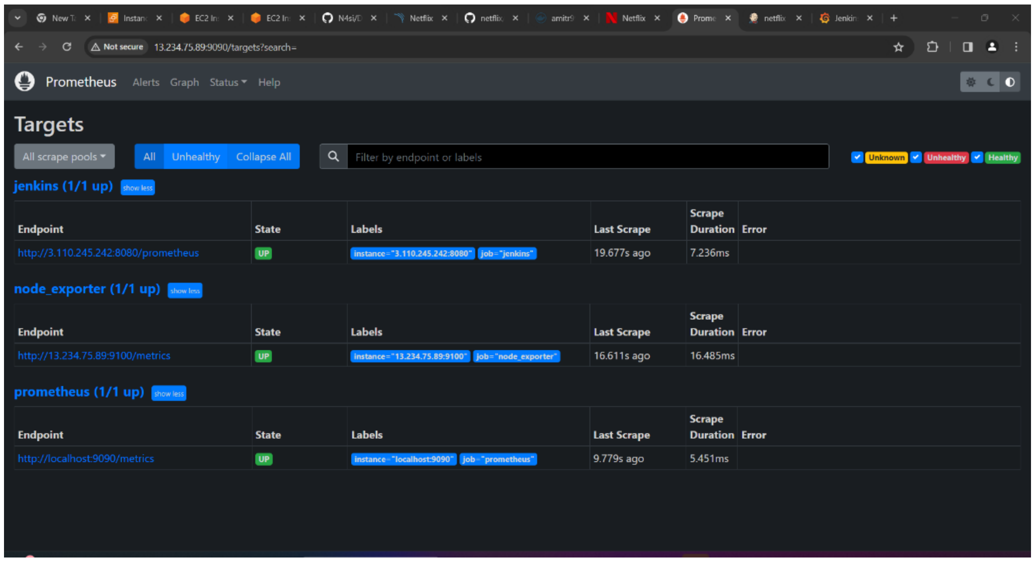
src/components/GenreBreadcrumbs.tsx
- Do not use Array index in keys — Code Smell, Major, Open, Not assigned, 5min effort — 1 year ago, L28 — jsx, performance, react

src/components/GridWithInfiniteScroll.tsx
- Do not use Array index in keys — Code Smell, Major, Open, Not assigned, 5min effort — 1 year ago, L54 — jsx, performance, react

src/components/VideoItemWithHoverPure.tsx
- Remove this commented out code. — Code Smell, Major, Open, Not assigned, 5min effort — 9 months ago, L33 — unused
- Remove this commented out code. — Code Smell, Major, Open, Not assigned, 5min effort — 9 months ago, L37 — unused

src/components/VideoPortalContainer.tsx
- 'pageYOffset' is deprecated. This is a legacy alias of `scrollY`. — Code Smell, Minor, Open, Not assigned, 15min effort — 1 year ago, L58 — cwe, obsolete

4.4 Prometheus



13.234.75.89:9090/targets?search=

Targets

jenkins (1/1 up)

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	

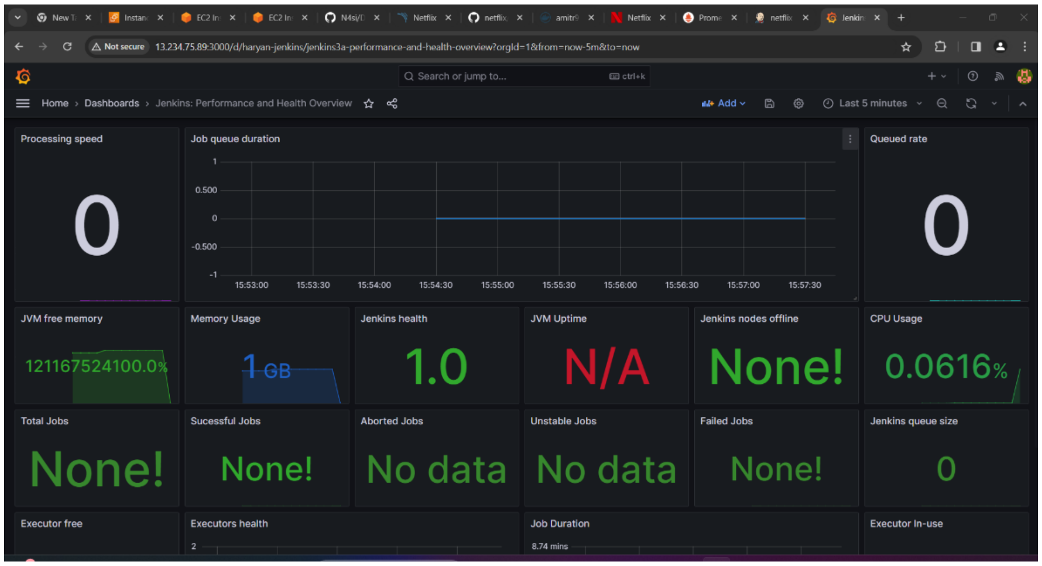
node_exporter (1/1 up)

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	

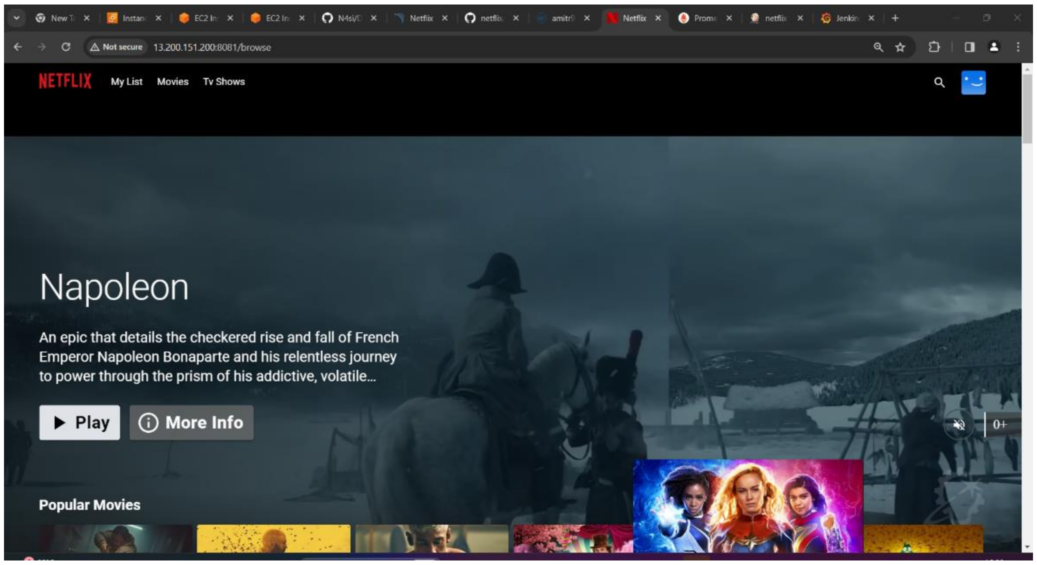
prometheus (1/1 up)

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

4.5 Grafana



4.6 Web Application Running



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...

5. CONCLUSION

5.1 Conclusion

Hence, we have successfully deployed a highly available and secure web server environment on Amazon Web Services (AWS). And ensured the reliability, performance, and security of the web application while maintaining efficient development and operational processes.

5.2 Future Scope

Docker is majorly considered as a best solution for service availability. It can be attached to implement continuous integration and continuous development i.e., CICD. It can be used in development when the software is getting develop this tool can used there to continuous security evaluation of software so that developers can program it more securely. As we have developed this tool for small scale. In future It can be used for large scale.

REFERENCES

Paper 1: Container orchestration and its implications on enterprise networking

Author: Mohanty, B., & Sarma, S.

Paper 2: Continuous Integration and Continuous Delivery in the IT Industry: A Case Study

Author: Stensholt, E., Landmark, A., & Fard, S. F.

Paper 3: DevSecOps: A Software Development Process for Securing Cloud-Native Applications

Author: Chapman, C., Muthanna, A., & Rajani, S.