Project Report on

Secure and Scalable CI/CD Pipeline with Docker, Kubernetes, Jenkins, Suricata, Promethius, and Grafana

Submitted by

Gaurav Bhise (240844223006)
Sartaj Mulla (240844223030)
Aditya Kulkarni (240844223022)
Jagdish Bhagwat (240844223005)
Sairaj Shinde (240844223042)

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:

Gaurav Bhise **(240844223006)**

Sartaj Mulla **(240844223030)**

Aditya Kulkarni **(240844223022)**

Sairaj Shinde **(240844223042)**

Jagdish Bhagwat **(240844223005)**

CERTIFICATE

This is to certify that the project report entitled **"Secure and Scalable CI/CD Pipeline with Docker, Kubernetes, Jenkins, Suricata, Promethius, Grafana"**, submitted by **Gaurav Bhise** 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

Guide

Mr. Vishal Salunkhe

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 **"Secure and Scalable CI/CD Pipeline with Docker, Kubernetes, Jenkins, Suricata, Promethius, Grafana"**, submitted by **Sartaj Mulla** 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

Guide

Mr. Vishal Salunkhe

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 **"Secure and Scalable CI/CD Pipeline with Docker, Kubernetes, Jenkins, Suricata, Promethius, Grafana"**, submitted by **Aditya Kulkarni** 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

Guide

Mr. Vishal Salunkhe

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 **"Secure and Scalable CI/CD Pipeline with Docker, Kubernetes, Jenkins, Suricata, Promethius, Grafana"**, submitted by **Jagdish Bhagwat** 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

Guide

Mr. Vishal Salunkhe

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 **"Secure and Scalable CI/CD Pipeline with Docker, Kubernetes, Jenkins, Suricata, Promethius, Grafana"**, submitted by **Sairaj Shinde** 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

Guide

Mr. Vishal Salunkhe

Course Coordinator

Mr. Nitin Kudale

CEO

Sunbeam Institute of Information Technology

Pune (M.S.) – 411057

APPROVAL CERTIFICATE

This Project report entitled **"Secure and Scalable CI/CD Pipeline with Docker, Kubernetes, Jenkins, Suricata, Promethius, Grafana"** by **Sartaj Mulla (240844223030)** 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 report entitled **"Secure and Scalable CI/CD Pipeline with Docker, Kubernetes, Jenkins, Suricata, Promethius, Grafana"** by **Aditya Kulkarni (240844223022)** 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 report entitled **"Secure and Scalable CI/CD Pipeline with Docker, Kubernetes, Jenkins, Suricata, Promethius, Grafana"** by **Jagdish Bhagwat (240844223005)** 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 report entitled **"Secure and Scalable CI/CD Pipeline with Docker, Kubernetes, Jenkins, Suricata, Promethius, Grafana"** by **Sairaj Shinde (240844223042)** 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 report entitled **"Secure and Scalable CI/CD Pipeline with Docker, Kubernetes, Jenkins, Suricata, Promethius, Grafana"** by **Gaurav Bhise (240844223006)** 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 modern cloud-native environments, ensuring secure, scalable, and automated application deployment is crucial. This project integrates **Docker, Kubernetes, Jenkins, and DevSecOps practices** to build a robust **containerized deployment pipeline** with security enforcement and real-time monitoring. The system follows a **multi-layered architecture**, incorporating **CI/CD automation, container orchestration, network security, and monitoring tools** to optimize software delivery.

Jenkins automates code integration, testing, and deployment with security checkpoints, while **Git** manages version control. **Docker** ensures consistency across environments, and **Kubernetes** handles workload management with auto-scaling and load balancing. **Suricata and IPTables** strengthen security by monitoring traffic and enforcing firewall rules, while **Prometheus and Grafana** provide real-time monitoring and alerting for performance and security insights.

The workflow begins with a code push triggering Jenkins, followed by containerization, security scans, automated testing, and deployment to Kubernetes. The system ensures **automated, secure, and scalable deployments**, reducing manual intervention while enhancing **threat detection, performance optimization**. This project demonstrates a comprehensive **DevSecOps-based deployment framework**, ensuring efficient and secure software delivery in dynamic cloud environments.

.

1. INTRODUCTION

In today's fast-paced software development landscape, Continuous Integration and Continuous Deployment (CI/CD) pipelines have become indispensable for accelerating software delivery, improving collaboration, and ensuring code quality. As organizations increasingly shift toward microservices architectures and containerized deployments, securing these pipelines becomes a top priority. A compromised CI/CD pipeline can introduce security vulnerabilities, leading to unauthorized access, data breaches, and service disruptions. This project, titled "Secure and Scalable CI/CD Pipeline with Docker, Kubernetes, Jenkins, Suricata, Prometheus, and Grafana," aims to establish a resilient DevSecOps framework that integrates security into the development lifecycle while maintaining efficiency and scalability. The objective is to automate application deployment while incorporating security measures to detect and prevent cyber threats and provide real-time monitoring of infrastructure health.

The project is designed to achieve several key objectives, beginning with the automation of the CI/CD pipeline using Jenkins to streamline build, testing, and deployment processes. Jenkins ensures seamless code integration, version control, and efficient application deployment through Kubernetes, which provides scalability, load balancing, and fault tolerance. Docker enables containerization, ensuring application consistency across environments, while Kubernetes orchestrates these containers, managing their scalability and self-healing capabilities. Security is a critical aspect of this implementation, with Suricata acting as an Intrusion Detection and Prevention System (IDPS) to monitor network traffic, detect anomalies, and mitigate threats.. IPTables and firewall rules further enhance security by filtering unauthorized traffic and preventing potential cyber threats.

To ensure performance monitoring and optimization, Prometheus is integrated into the pipeline to collect and store real-time system and application metrics. These metrics are then visualized through Grafana, providing real-time dashboards that display critical information such as CPU usage, memory consumption, network traffic, and security alerts. Alerts and notifications are configured to enable proactive incident response in case of failures or security breaches. Scalability and high availability are achieved through Kubernetes' dynamic workload management, ensuring that applications are deployed across multiple nodes for fault tolerance and resilience.

Each technology in this project plays a crucial role in securing and optimizing the deployment pipeline. Jenkins automates CI/CD tasks, Docker ensures container portability, Kubernetes manages orchestration, Suricata provides network security, Prometheus collects performance metrics, Grafana visualizes monitoring data, and IPTables enforces network security policies. The implementation follows a structured approach, starting with the setup of the CI/CD pipeline, where Jenkins is configured for automated builds, tests, and deployments, incorporating security checkpoints at each stage. Containerized applications are then deployed using Kubernetes. Security hardening measures include configuring Suricata for network monitoring. Monitoring and logging solutions are

implemented through Prometheus for metric collection and Grafana for dashboard visualization, while logging and auditing mechanisms ensure traceability.

This project presents a comprehensive approach to securing and scaling CI/CD pipelines by integrating automation, security, and monitoring. By leveraging Jenkins, Docker, Kubernetes, Suricata, Prometheus, and Grafana, it establishes a robust DevSecOps framework that aligns with modern cybersecurity and DevOps best practices. Organizations adopting this solution can achieve faster, more secure, and highly available application deployments while proactively detecting and mitigating security threats. By embedding security into every stage of the pipeline, this approach enhances the software development lifecycle (SDLC), ensuring resilient and reliable software delivery in an increasingly complex and security-conscious environment.

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: - Deploying Jenkins, Ansible and Kubernetes to Automate Continuous Integration and Continuous Deployment Pipeline

Author: Neelam Singh, Aman Singh, Vandana Rawat

Description: A pipeline is set up to build code, run tests, and securely deploy a newer version of the application. Manual errors are minimized or can be eliminated through automated pipelines thus providing developers with standardized feedback loops and facilitating rapid product iteration. This article introduces the entire automated CI/CD channel by deploying Jenkins, Ansible and Kubernetes cluster to host the updated version. The proposed approach provides an easy to implement automation methodology with high scalability and resiliency. The study aims to provide an approach for continuous integration (CI), continuous testing, and continuous delivery (CD) for automating compilation of source code, analysis, testing, deployment, and reports using build pipeline model.

Paper 2: - Automation of Microservices Application Deployment Made Easy By Rundeck and Kubernetes

Author: Harika Rajavaram, Vineet Rajula, B. Thangaraju

Description: Today, more than 2.6 million apps are managed on Google Play Store [1]. To be able to compete with this type of huge volume of applications, a powerful, easy to scale and a flexible application is required to sustain in the market and overpower the competitors. Microservices architecture and continuous deployment practices tackle the above challenges and help to improve the productivity of the organizations. This paper explores the different procedures of deploying microservices in CI/CD pipeline using Rundeck, Docker containers, Docker-Compose and Kubernetes. Moreover, it weighs the pros and cons of these orchestration methods, explaining suitable use cases for each one of them. It concludes by discussing that Kubernetes is the most efficient way to deploy microservices to achieve high availability and scalability.

Paper 3: - Kubernetes Continuous Deployment

Author: Sakshi Sharma, Ayush Shankar Pathak, Harshit Malhotra, Garima Pandey

Description: The scope of this research paper is to develop a continuous integration/continuous development pipeline for microservices application, decide which options will include in our continuous integration phase, utilization of Jenkins to execute blue/green or moving sending, utilizing Ansible or integration/continuous deployment, is the foundation to assemble our "framework"; i.e., Kubernetes Clusters and to make our undertaking stick out, we have actualized checks, for example, security examining, execution testing, joining testing, and so forth! Henceforth, AWS CloudFormation licenses you to exhibit your entire establishment and application resources with either a book record or programming vernaculars. The AWS CloudFormation Registry and CLI simplify it to manage untouchable resources with CloudFormation. This gives a lone wellspring of truth for all of your resources and urges you to standardize system parts used across your affiliation, engaging arrangement consistence, and faster examining. CloudFormation manages choosing the right errands to perform while managing your stack, orchestrating them in the most capable way, and moves back changes normally if botches are recognized. On the consummation of continuous integration, we have set up the continuous deployment which incorporates pushing the constructed Docker container(s) to the Docker storehouse and deploying these Docker container(s) to a little Kubernetes bunch. For our Kubernetes cluster, we have used AWS Kubernetes as a service; to deploy our Kubernetes cluster, we used cloud formation tool on AWS. The languages used for the deployment are PHP and JSON. Preferably, we have been able to run these from within Jenkins as an independent pipeline.

3. SYSTEM DEVELOPMENT AND DESIGN

3.1 Proposed System

The proposed system follows a multi-layered architecture designed to enhance security, scalability, and automation in software deployment. It integrates Continuous Integration and Deployment (CI/CD), containerization, orchestration, security enforcement, and monitoring to ensure an efficient DevSecOps workflow. Jenkins, configured as the CI/CD automation server, streamlines code integration, testing, and deployment with multi-stage pipelines that include security checkpoints, automated unit testing, and static code analysis. Git is integrated for version control, enabling rollback capabilities and ensuring efficient change management. Jenkins also exposes monitoring metrics at /prometheus for real-time CI/CD performance tracking.

To ensure environment consistency and efficient resource utilization, Docker is used for containerization, packaging applications into lightweight, isolated containers. Kubernetes orchestrates these containers, providing auto-scaling, load balancing, rolling updates, and self-healing mechanisms. Additionally, Suricata, an Intrusion Detection and Prevention System (IDPS), continuously monitors network traffic for threats, providing real-time alerts for anomalies.

Monitoring and logging play a crucial role in system performance and security. Prometheus is deployed to collect real-time system and application metrics, with Node Exporter integrated to track host-level metrics such as CPU, memory, disk, and network usage. Prometheus is configured as a systemd service for automated startup, with predefined targets monitoring Kubernetes nodes, application services, and CI/CD pipelines. Grafana is installed for advanced data visualization, configured with Prometheus as a data source. A predefined dashboard (Dashboard ID: 1860) visualizes key performance indicators, including resource utilization, network activity, and container performance. Jenkins is also monitored by integrating the Prometheus Plugin, exposing CI/CD pipeline metrics, which are visualized in Grafana using Dashboard ID: 9964.

The system workflow is structured to ensure a seamless and secure deployment process. Developers push code to Git, triggering the Jenkins pipeline, which builds the application

into a Docker container. Then container is deployed to the Kubernetes cluster. Prometheus continuously collects system metrics, while Grafana provides insights into system performance, and Suricata detects potential threats. Any security anomaly triggers alerts, and firewall rules ensure unauthorized access is restricted.

By integrating CI/CD automation, containerization, security enforcement, and real-time monitoring, the proposed system creates a robust, scalable, and secure software deployment environment. It ensures efficient application delivery while proactively mitigating security risks, making it an ideal solution for cloud-native architectures.

.

3.2 Flow chart

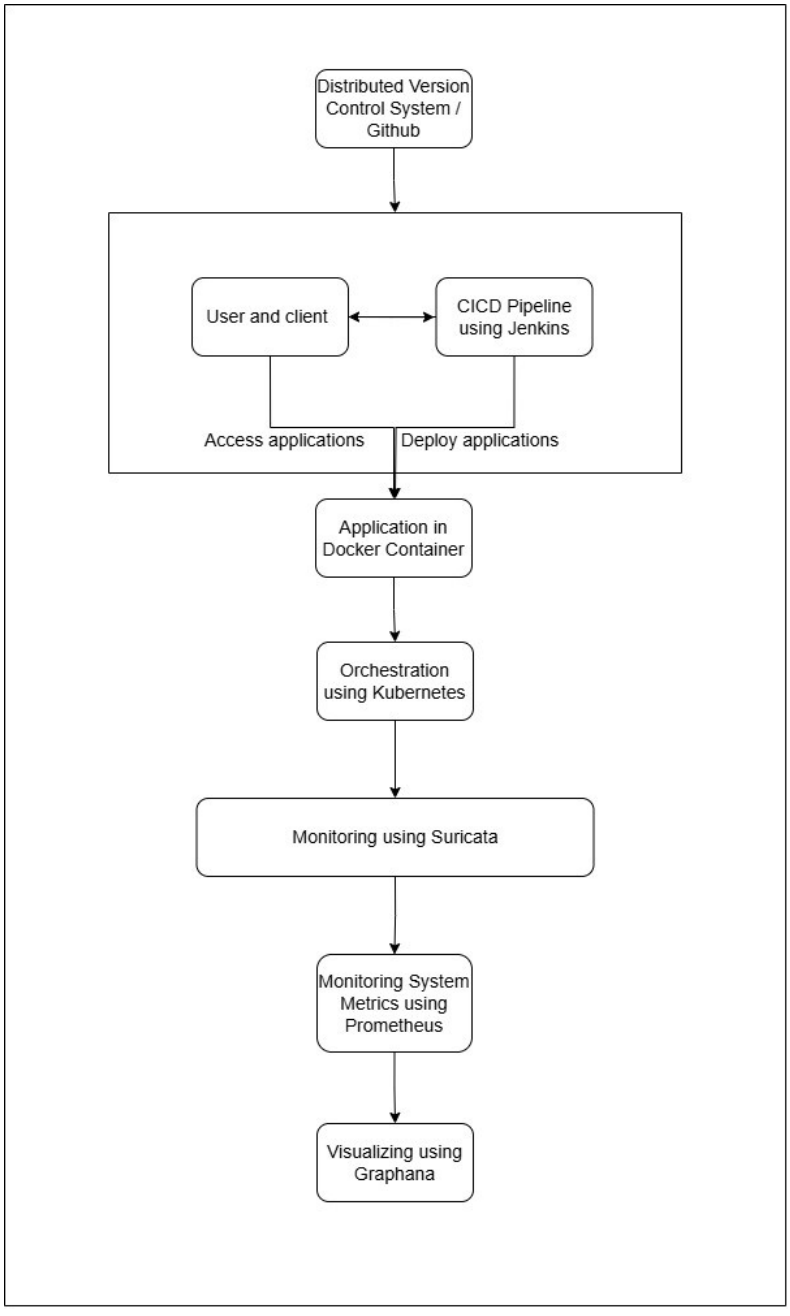


Figure: Flowchart

3.3 Technology used

3.3.1 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.

3.3.2 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.

3.3.3 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.

3.3.4 Kubernetes

Kubernetes is an open-source container orchestration platform designed to automate the deployment, scaling, and management of containerized applications.

Key features of Kubernetes:

Container Orchestration: Manages the deployment and scaling of containerized applications across a cluster of machines, ensuring high availability and resource efficiency.

Auto-scaling and Load Balancing: Automatically scales applications based on demand and distributes network traffic to maintain performance.

Self-healing: Detects and replaces failed containers, ensuring application stability and continuity.

Service Discovery and Networking: Facilitates seamless communication between services within the cluster using built-in DNS and network policies.

Declarative Configuration: Uses YAML or JSON configuration files to define and manage infrastructure, enabling automation and consistency.

3.3.5 Prometheus

Prometheus is an open-source monitoring and alerting toolkit designed to collect and analyze time-series data from applications and infrastructure.

Key features of Prometheus:

Time-Series Data Collection: Continuously scrapes metrics from configured targets and stores them in a time-series database.

Powerful Query Language (PromQL): Enables users to analyze and aggregate metrics for performance monitoring and troubleshooting.

Alerting Mechanism: Integrates with Alertmanager to send real-time alerts based on predefined conditions.

Service Discovery: Automatically detects targets in dynamic environments such as Kubernetes, reducing manual configuration.

Scalability and Reliability: Designed for high availability, Prometheus runs independently without external dependencies.

3.3.6 Grafana

Grafana is an open-source visualization and analytics tool used to create interactive dashboards for monitoring system performance and security metrics.

Key features of Grafana:

Customizable Dashboards: Provides rich visualizations of time-series data with graphs, charts, and heatmaps.

Multi-source Data Integration: Supports various data sources, including Prometheus, InfluxDB, Elasticsearch, and MySQL.

Alerting System: Sends notifications via email, Slack, or other channels when predefined thresholds are met.

User Access Control: Allows role-based access management to secure dashboard configurations.

Plugin Support: Extensible through plugins, enabling integration with third-party tools and applications.

3.3.7 Suricata

Suricata is an open-source network threat detection system that provides intrusion detection, intrusion prevention, and network monitoring capabilities.

Key features of Suricata:

Intrusion Detection and Prevention (IDS/IPS): Analyzes network traffic in real time and detects malicious activity based on predefined rules.

Deep Packet Inspection: Inspects network packets at multiple protocol layers to detect threats and anomalies.

Threat Intelligence Integration: Supports external threat intelligence feeds to enhance detection capabilities.

High Performance and Scalability: Utilizes multi-threading to efficiently process large volumes of network traffic.

Logging and Reporting: Generates detailed logs and alerts for security analysis and compliance.

3.3.8 IP Tables

IP Tables is a Linux-based firewall tool used to configure and manage network traffic filtering and security rules.

Key features of IP Tables:

Packet Filtering: Controls inbound and outbound network traffic based on predefined rules.

Network Address Translation (NAT): Enables translation of IP addresses and ports for secure and efficient communication.

Chain-based Rule Management: Organizes rules into chains for input, output, and forwarding of packets.

Logging and Auditing: Captures logs of network activity for security monitoring and forensic analysis.

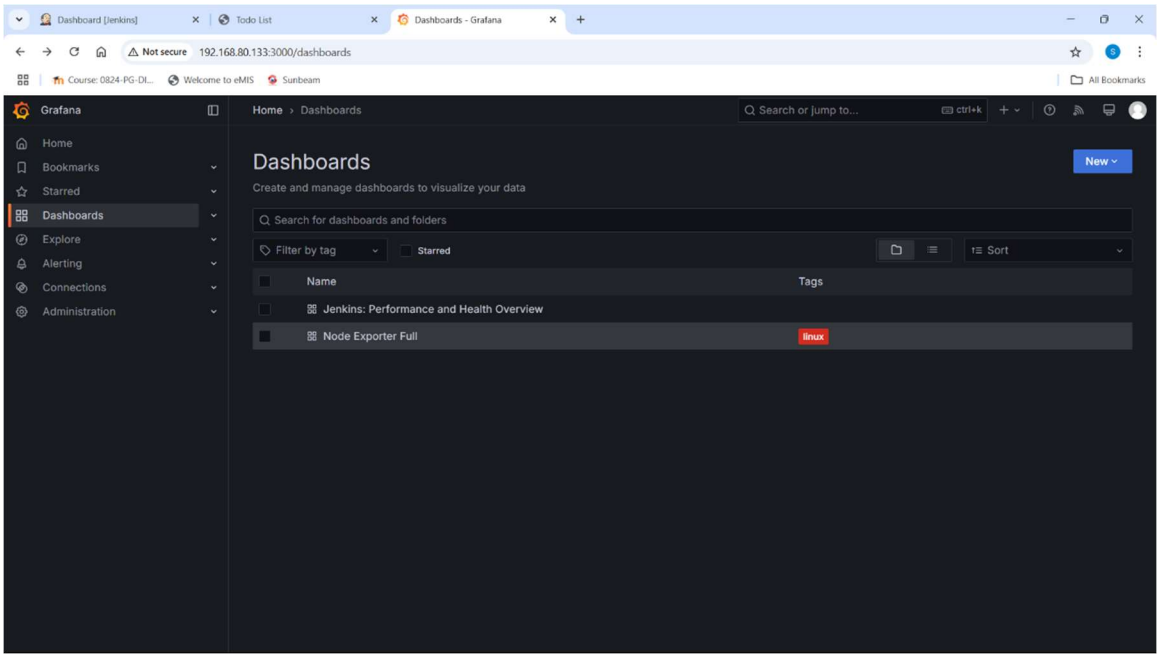
Integration with Security Policies: Works alongside other security tools like Suricata to enforce access controls and prevent unauthorized traffic.

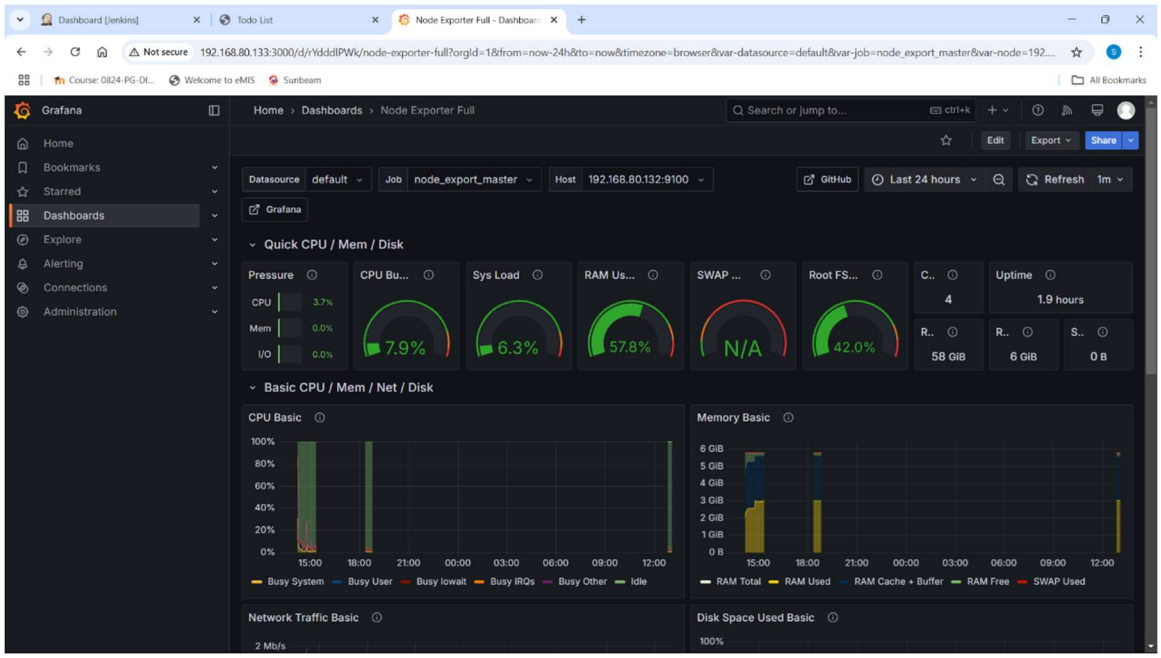
This technology stack ensures robust security, automation, and monitoring for modern infrastructure deployments.

4. Project Output

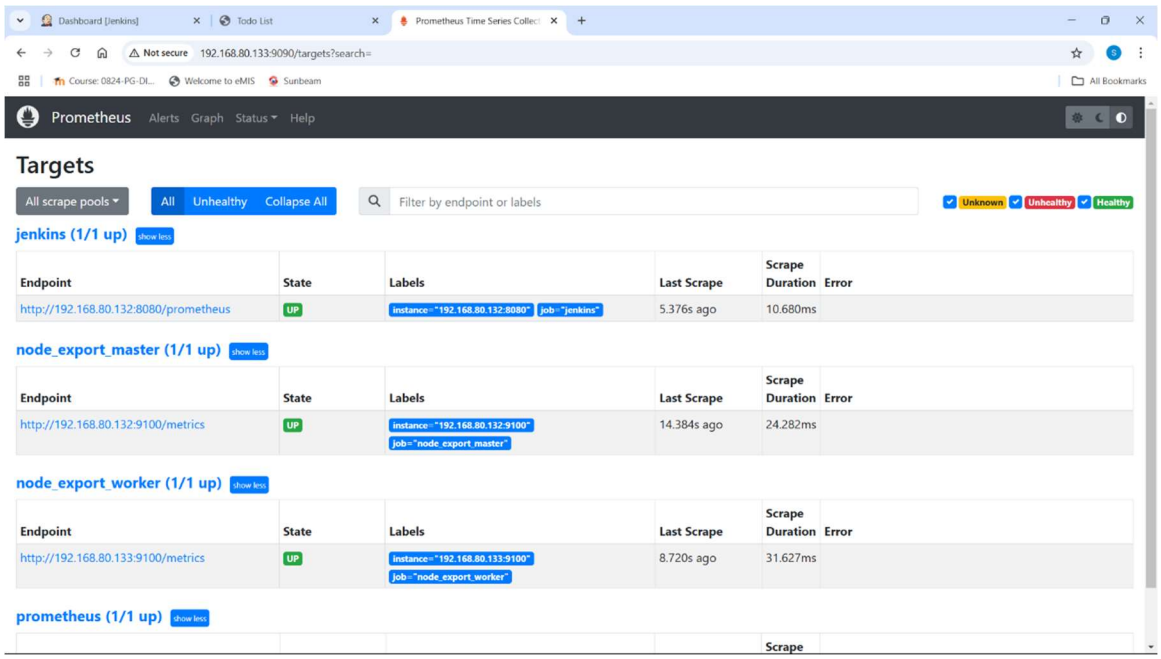
4.1 Web Application

4.2 Grafana

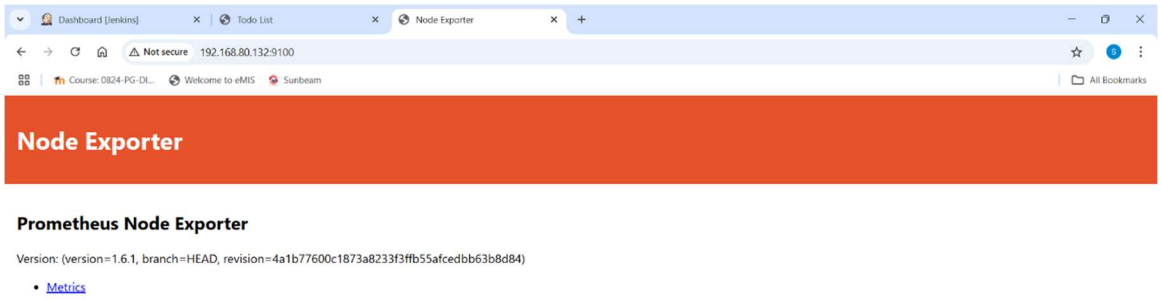




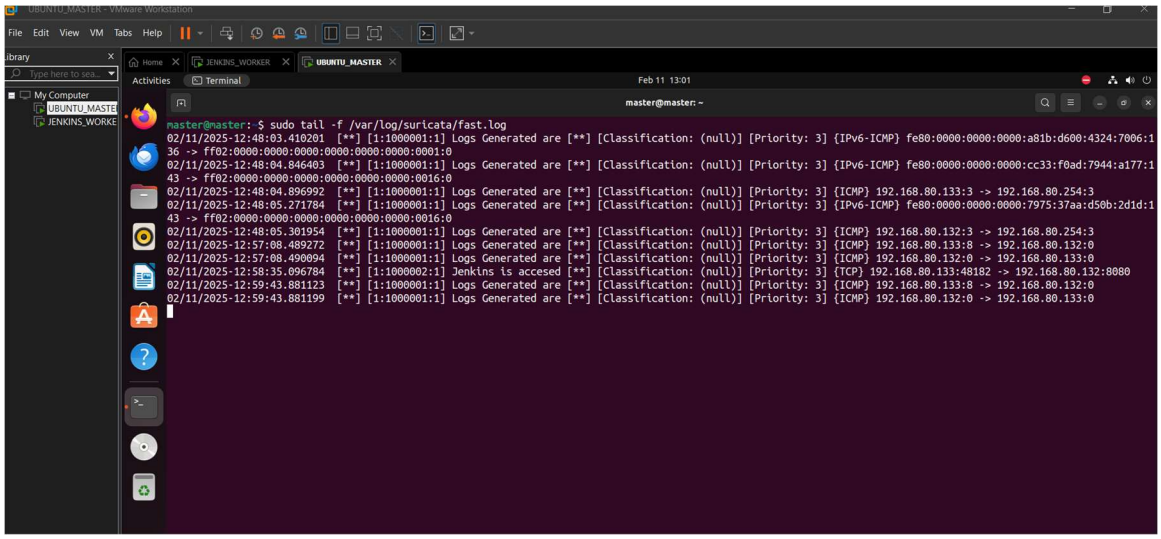
4.3 Promethius



4.4 Node Exporter



4.5 Suricata



5. CONCLUSION

5.1 Conclusion

This project successfully implements a **secure, automated, and scalable** containerized deployment pipeline using **Docker, Kubernetes, Jenkins, and DevSecOps practices**. By integrating **CI/CD automation, security enforcement, and real-time monitoring**, the system ensures **efficient software delivery with minimal manual intervention**. Security tools like **Suricata and IPTables** strengthen network protection, while **Prometheus and Grafana** provide real-time performance and security monitoring. The use of **Kubernetes** ensures high availability, scalability, and efficient resource management. Overall, this project demonstrates a **robust and resilient DevSecOps-driven deployment framework**, enabling organizations to streamline their software deployment while ensuring compliance and security.

REFERENCES

Paper 1: - Singh, N., Singh, A., & Rawat, V. (2022, December). Deploying Jenkins, Ansible and Kubernetes to Automate Continuous Integration and Continuous Deployment Pipeline. In *2022 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI)* (pp. 1-5). IEEE.

Paper 2: - Rajavaram, H., Rajula, V., & Thangaraju, B. (2019, July). Automation of microservices application deployment made easy by rundeck and kubernetes. In *2019 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)* (pp. 1-3). IEEE.

Paper 3: - Sharma, S., Pathak, A. S., Malhotra, H., Ghosh, S., & Pandey, G. (2022, January). Kubernetes Continuous Development. In *Proceedings of International Conference on Recent Trends in Computing: ICRTC 2021* (pp. 559-568). Singapore: Springer Nature Singapore.