BUILDING A COMPREHENSIVE SECURED CI/CD PIPELINE WITH INTEGRATED MONITORING AND ALERTING SYSTEM

JOY MIAH

ID: 212-15-4108

This Report Presented in Partial Fulfilment of the Requirements for the Degree of

Bachelor of Science in Computer Science and Engineering

Supervised By

DR. SHEAK RASHED HAIDER NOORI
Professor & Head
Department of CSE
Daffodil International University

Co-Supervised By

MD. SHOHIDUL ISLAM POLASH
Lecturer
Department of CSE
Daffodil International University

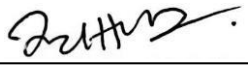
DAFFODIL INTERNATIONAL UNIVERSITY
DHAKA, BANGLADESH
JULY 2024

APPROVAL

The project entitled " **Building a Comprehensive Secured CI/CD Pipeline with Integrated Monitoring & Alerting System**", presented by **Joy Miah** (ID: **212-15-4108**) to the Department of Computer Science and Engineering at Daffodil International University, has been deemed satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Engineering and has been approved in terms of its style and content. The presentation took place on 16th July 2024.

BOARD OF EXAMINERS

Dr. Sheak Rashed Haider Noori (SRH) — **Board Chairman**
Professor & Head
Department of CSE
Faculty of Science & Information Technology
Daffodil International University



Dr. Md. Zahid Hasan (ZH) — **Internal Examiner 1**
Associate Professor
Department of CSE
Faculty of Science & Information Technology
Daffodil International University

Mr. Abdus Sattar (AS) — **Internal Examiner 2**
Assistant Professor
Department of CSE
Faculty of Science & Information Technology
Daffodil International University

Dr. Md. Manowarul Islam (MI) — **External Examiner**
Associate Professor
Department of CSE
Faculty of Science & Information Technology
Jagannath University

DECLARATION

We solemnly affirm that this project results from our efforts, guided by **Dr. Sheak Rashed Haider Noori**, Head of the Department of CSE at Daffodil International University. We hereby reiterate that this project or any part of it has not been submitted for any award of any degree or certification in another institution.

Supervised by:

Dr. Sheak Rashed Haier Noori (SRH)
Professor and Head
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Co-Supervised by:

Md. Shohidul Islam Polash (SIP)
Lecturer
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Submitted by:

Joy Miah
ID: 212-15-4108
Department of CSE
Daffodil International University

ACKNOWLEDGMENT

I want to thank the almighty for the numerous privileges and mercies, for it was also very helpful to me to complete my final year project successfully.

I feel honored to be under the supervision of **Dr. Sheak Rashed Haider Noori**, a brilliant and renowned professor and head of the Department of Computer Science & Engineering at Daffodil International University. I would also like to thank all the educators and the staff members who assisted me whenever I required their assistance in any segment of my undertaking. Furthermore, I'm so grateful to all the teachers of Daffodil International University for the kind of environment they have provided to build us.

I am grateful to **Dr. Sheak Rashed Haider Noori**, the professor and head of the Department of CSE at Daffodil International University, for his kind permission and help regarding my project.

Finally, special gratitude goes to my family members, friends, colleagues and well-wishers for their continued support and motivation throughout the process. All the good-hearted people who gave me this opportunity without whom I couldn't have done it.

ABSTRACT

The rapidly evolving landscape of software development is a rather young and actively developing field. There is a tendency for organizations to provide efficient and high-quality applications. This project involves creating a fully automated continuous integration/continuous delivery pipeline that would seek to make application delivery as automated as possible with very little to no human disruption, with frequent updates of new features and releases to the customers.

I have developed a continuous integration/continuous delivery pipeline that helps to simplify the deployment chain thereby helping shorten the amount of time it takes to deploy an application. I also create and deploy monitoring tools that give information as to how well an application is running and its condition in real time. I have done configuration of proactive alerts based on predefined rules to enable swift issue resolution & efficient problem-solving. It is possible to keep the code quality high and detect security issues at early stages by integrating security checks and automated testing.

This project also provides high availability and scalability. This system can run applications uninterruptedly and smoothly. They can scale up and scale down based on requests by utilizing Kubernetes. I have also incorporated automated rollbacks, which enhance the stability of the deployment process by allowing for quick reversion in case of any deployment failures.

TABLE OF CONTENTS

LIST OF FIGURES/SCREENSHOTS

CHAPTER 1

INTRODUCTION

1.1 Introduction

In the advanced generation of software development, it becomes a challenge for organizations to meet the stake holder's expectations of providing quality applications that are developed and delivered within the shortest time possible and in a reliable manner.

The fact that these processes should be as efficient as possible because the involvement of staff members is likely to slow down the process or introduce certain mistakes. Particularly, this project's main concern on the development and improvement of the CI/CD pipeline to keep on automating the deployment procedure which will release updated versions of the software to the end-users more frequently and efficiently.

One of the key features of this project is to have an efficient CI/CD process so that the application deployment process can be made easy and quick. Also, the application of performance management and monitoring tools along with real-time alert systems provide regular updates on application performance and quick resolution of problems. Therefore, the planned integration of security checks and automatically run tests is to ensure the nearest control of code quality and its security Stage.

Availability and scalability are also considered important. With Kubernetes applications can be scaled effectively and run without issues. Automated rollbacks also increase the robustness of the deployment since they enable a fast rollback in occurrences of failures. This pipeline accelerates and increases in the quality of application delivery but also guarantees that the final product will be optimized for performance and protected from security threats.

1.2 Motivation

Others are meeting difficulties in growing large-scale applications as well as the challenges associated with load balancing and highly available systems that influenced my project. I study computer science, so I'm always interested in situations like when Amazon has a christmas sale, when there's a sudden spike in user traffic from one million to ten million. It's difficult to control such much traffic while maintaining a positive customer experience. Furthermore, even a minute of downtime can have a significant effect on the company. How infrastructure specialists handle these circumstances and accomplish a rollback when necessary piques my curiosity. How this type of solution was provided by the solutions architect.

Additionally, when managing a large infrastructure, it’s really important to have real-time insights into how the system is doing. This is a similar concept to detecting and quickly rectifying an adverse failure condition if one of the nodes or components folds. This has inspired me mainly by receiving the railway application during the ‘Eid’ Occasion and the ineffectiveness of the servers during the result publication in Bangladesh. I would be willing to work on solutions for these cases. If you don’t have a good hosting service, then you are toast. Concerning such things, here is the following project under integrated monitoring and alerting. Technological ones, as well as the development of more sophisticated CI/CD processes for efficiently deploying applications, took place. A focus on enhancing the reliability and performance of large-scale system technologies will be made. The idea is to make them capable of handling large client loads while maintaining maximum availability and extensibility.

1.3 Objectives

The core objective of our proposal is to define an enhanced CI/CD framework with an embedded Kubernetes cluster, monitoring & alerting system, high availability, scalability & security. The first is to create a solid, effective, and reliable CI/CD pipeline for software applications to increase the effectiveness and accuracy of the application deployment cycle.

The technical objectives of our project include:

Holistic CI/CD Pipeline: The goal is to have one end-to-end CI/CD scheme with all the features of good DevOps that helps with design, testing, vulnerability assessment, development of the applications and deployment.

Intuitive Interface: Our challenge here is to create a simple, clear interface by in order to help the developers and system administrators to have the smooth interaction with the system. The interface should help control the deployment process, view data in real time and easily navigate various operations.

Real-Time Monitoring & Alerting: In this regard, I anticipate adding functions that allow real-time monitoring. In this strategy, all the infrastructural and network metrics are collected and maintained. I've configured alerts through the help of Prometheus, Alert-Manager, and Grafana. This includes monitoring the performance and health of applications. Providing a way of detecting problems and rectifying them before they become uncontrollable.

Reliable Application Deployment: The goal is to establish a stable system that consumes the minimum amount of system resources and time as possible. Deployment system that would simplify the handling of the actual deployment of the project. This includes uploading the built application image to the respective docker repository and the deployment for applications on a Kubernetes cluster via ArgoCD.

Fault Tolerance: It involves developing a system that can be tolerant of faults and perform consistently and independently. Communicative properties correctly functioning of the application despite of presence of faults. These include managing faults in the deployment process. The availability of the application that is supposed to be deployed and making sure that there is a way of recovering from the failures mechanisms.

Scalable Deployment: The next step that intend to undertake is creating a scalable deployment system through a K8s cluster. That enables users to deploy applications on a larger scale and with increased availability. This includes features such as load balancing and automatic scalability as well as other features added as part of service management.

1.4 Expected Outcomes

From the performance appraisal this end-to-end Implementation of an automated software development pipeline with a special focus on the monitoring & alerting system. Develop an application that can be deployed on a Kubernetes Cluster with high availability & scalability. Among these anticipated results are:

Effective Application Deployment: The deployment of the CI/CD pipeline will only require minimal human interaction as it covers all the areas of application deployment. It will make the subsequent deployments much quicker and release developers and DevOps to concentrate on other areas such as coding and product innovation.

Time and Cost Efficiency: This will solve a problem by significantly reducing the amount of time and energy required for deploying the application. Thus, organizations will be able to reduce operation costs and eliminate any wastage within the resource management cycle that will lead to increased profit margins based on lower costs.

Enhance Code Quality and Security: Such integration of the automated security checks and testing will ensure that the code quality is top notch and that any security issues are identified early in the development process. Such an approach that will assist in keeping applications secure and steady, and therefore serve a proactive strategy.

Real-Time Monitoring and Proactive Alerting: The use real-time monitoring tools will ensure about the status of the applications and their functioning in the systems environment.
Analysis of alerts that are predicted using predefined rules will allow efficient problem solving and will ensure that the system is running optimally with as little disruption as possible.

High Availability and Scalability: With the help of Kubernetes, which make sure that application can easily scale up and scale down based on traffic. This ensure scalability and high availability.

Stability through Automated Rollbacks: The use of automated roll back will improve the reliability during the process of deployment. If there are any failures in deployment quick reversions will minimize the impact on end-users and maintain application reliability.

Efficient Resource Utilization: The project will maximize resource consumption by self-provisioning of infrastructures. This will make the process more efficient and save costs and properly distribute the development resources.

1.5 Project Management and Finance

In project management, accurate prediction of the costs comes on the need while doing this building a CI/CD pipeline for automation with an integrated monitoring & alerting system. The software is developed for academy usage as part of the web-based application on a Kubernetes cluster with high availability & scalability. Together, they ensure that the system will be developed and implemented with the least amount of the required financial resources. When implementing this project, consider the following important policies on project management as follows:

Establish Project Scope: Explain the purpose, goals, and outcomes of the proposed effort in a precise manner. Identify the key requirements and functionalities that the designed system should have. This will help in exactly fulfilling the requirements of people.

Distribute Resources: Prioritize all the needed resources for the project. In making that, it simply means the project should possess that knowledge and overcome difficulties and the process work.

Create Budget: Determine the possible cost that may be incurred when developing and implementing the system. These could range from the cost of developing software to the cost of hardware acquisition, and the cost of the licenses. Carry out a comprehensive well-analyzed budget.

Financial Forecasting: Pre-design the overall monetary profit or loss that accompanies the particular system's implementation. Take into substantial & account elements such as improved efficiency outcomes and reduction in Labor cost. To provide assessment of the amount of return on investment that will help with the promotion of the project. The things you need to watch in cost consideration include in assessing the financial feasibility you should observe the work that has been done with the budget and the timeline set for the project. Stakeholders are informed in

communication management which deals with stakeholder updates, notifications, and feedbacks. The use of continually organizing meetings and reporting progress and follow-up activities can help prevent repeat occurrences.

Risk management: This stands to avoid the process of identifying possible risks and planning. What should be done to minimize such risks use risk reduction measures on the project. This project also have the desire to contribute to the even more advancement of DevOps that will complement cloud-native applications to build an end-to-end CI/CD pipeline with integrated high availability.

1.6 Report Layout

The project can be titled as creation of the overall CI/CD pipeline with multiple monitoring and alerting systems. This specifies the system for deploying applications on a Kubernetes cluster with high availability and scalable environment.

Chapter-1: Introduction

In this chapter on communication, the reader will find information about the objectives of the project, inspirations, and anticipated outcomes with a brief description of the work.

Chapter 2: Background

Here the reader will find information about the procedure of decision-making, the planning approach, the background of the project under consideration and justifications.

Chapter 3: Requirement Specification

This chapter describes the necessary hardware and software for the project and the general work. One must present all related specifications, knowledge, and skills needed to complete the project.

Chapter 4: Design Specification

This chapter extends the description of the system architecture to the extent and gives detailed consideration of the implementation of the various activities relative to system development and requirement implementation.

Chapter 5: Implement and Testing

This chapter highlights the process involved in developing pipelines. Particularly at the installation and testing stages. The process passes in this way is error testing after passing the stages of

debugging and validation. The system is fully functional as it should be as how it is expected or how it is supposed to be.

Chapter 6: Impact on Society

In this chapter, briefly describes some of the impacts or could have, brought about the aspect of sustainability to the environment and the specific development industry. Implication of the above system adopted in such existing organizations.

Chapter 7: Conclusion and Future Scope

I would like to focus on the final aspect of the report which is the future of the project and its further utilization. It is about the goals of the study and the key findings and contributions.

CHAPTER 2

BACKGROUND

2.1 Preliminaries and Terminologies

The project entitled '**Building a Comprehensive CI/CD Pipeline with Integrated Monitoring & Alerting System: Deploy Applications on Kubernetes Cluster with High Availability and Scalability**' is a complete software solution offered for the software development life cycle. Regarding the problems, there is a variety of modules that are intended to function in the areas of building, testing, and performing static code analysis for applications before the application deployment. The overall significance of this project is in the concept of betterment of the business by refining its operational efficiency in terms of deployment and enhancement in the software applications. By centralizing and automating these core processes, the system provides the developers and system administrators the power to monitor and manage key metrics, operations, and performance in real-time and make informed decisions regarding. This would be possible with a project that will use Jenkins for automation of continuous integration and SonarQube for static code analysis, ArgoCD for declarative designing and running of infrastructure and applications, Prometheus & Grafana for monitoring and alerting. These technologies provide an ideal uninterrupted and optimized CI/CD mechanism for software applications. This method can save the way to the future advancement of how application deployment is being done today. This project will create a way for further improvements in the future.

2.2 Related Works

Numerous field projects and programs have provided the groundwork for the concept of constructing a holistic CI/CD framework with integrated monitoring & alerting. Which is a great framework for deploying applications on the Kubernetes cluster with high availability and scalability.

Jenkins in DevOps: Jenkins is a widely used open-source automation server that aids developers in the consistent development, testing, and deployment of software. It makes it simple to build up an environment for continuous integration and development for almost any set of languages and Source code repositories.

SonarQube software quality management: SonarQube is an open-source tool for continuous code quality assessment. It automatically analyzes the code using static analysis to identify mistakes, poor code, and security vulnerabilities.

GitOps ArgoCD: ArgoCD is a Kubernetes continuous delivery solution that is declarative. It follows the GitOps principles according to which git repositories should be used as the source when defining the desired application state.

Prometheus and Grafana in Observation: Grafana, a multi-platform open-source analytics and Interactive visualization web application. Prometheus is an open-source systems monitoring and alerting toolkit that is frequently combined to enable real-time software application monitoring.

Application Deployment: Kubernetes is a solution based on open-source that provides for automation of the application containers deployment, scaling, and running. These projects works directly with Jenkins, Kubernetes, SonarQube, ArgoCD, Prometheus and Grafana by updating the source code. To ensure scalable and efficient deployment continuous integration and continuous delivery refers to the use of automation tools and processes which allow for the integration of code changes and the continuous delivery of updated products to consumers.

2.3 Comparative Analysis

Some things differentiate this project from traditional approaches to the development and deployment of software. End-to-end continuously integrated and continuously delivered software utilizing the mechanisms of the described CI/CD pipeline. I have developed in private workspace or data center where I have to do lots of manual stuffs such as configuring software load balancer, network plugin e.tc. Also, there is an exceptional element in our pipeline which means that the integrated monitoring and alerting option neutralizes a large number of resource-consuming. This puts my project in the customization of such a service of any organization. First of all, Jenkins, Travis CI, and other similar services are more generalized in the case of the CI/CD function whereas our project is much closer to the complete action flow from the programming of the pipeline to its monitoring and alerting. Feedback on the utilization of applications is offered instantly across all the explored channels. It specifically focuses on the integration of Jenkins, SonarQube, Argo-CD, Prometheus, and Grafana to have robust CI/CD pipelines. Both Jenkins and Travis CI include the performed analytics with customizable dashboards. The presented project

will be driven by specific technologies to pull analytics on a real-time diversity of performance data from the implemented application. Concerning data security both platforms claimed to be important. This project properly implemented secure deployment processes and storage mechanisms respectively.

2.4 Scope of the Project

The Interoperability of services will also be addressed followed by effective communication and data exchange. This will be based on the implementation of standardized protocols and data formats. Together with the integration of middleware that will help communicate between the services. The other critical aspect of the project will concern its resiliency. This can be realized by applying fault-tolerant design principles in redundancy, failover, and gracefulness in the event of failures. Degradation, the system shall be designed in such a way that it remains unaffected by the failure of an individual service. Another important aspect in the realization of the project will be scalability. Load-balancing techniques will make it possible to add or remove resources if needed. Monitoring and logging also form part of the project. Any activities within the service delivery environment shall be fully traceable through a comprehensive system of monitoring and logging. Provided here is the visibility into system performance and in case anything goes wrong, it will help in fast identification and troubleshooting.

2.5 Challenges

Data Integrity and Consistency: It becomes very tough to maintain data integrity as well as consistency between all the services. As a result, this happens because micro-services are distributed architecture and communication may happen asynchronously.

Scalability: The system is expected to scale properly on the basis of increasing data. It requires proper planning and load-balancing techniques. Sometimes it needs adding or removing of resources at times.

Fault Tolerance: This would be somewhat challenging to implement. To make it fault-tolerant, it must be resilient upon faults through redundancy, failovers and graceful degradation. So that the failure of specific services handling this information does not affect the program everywhere.

Security: Personal information must be very confidential while in transmission. Powerful security measures such as encryption and access controls have to be put in place to safeguard the entire system.

Interoperability: Because different services must communicate and exchange data successfully. Standardized protocols and data formats must be utilized.

Real-Time Analytics: With a view to supporting real-time analytics that will enable better business decisions and an efficient data streaming pipeline. Ensuring that these data are correct and timely may be a big challenge.

Monitoring and logging: Having a good monitoring and logging system in place may help you resolve issues more easily and provide great insight into how the system works. Some difficulties are there in developing an end-to-end CI/CD pipeline with integrated monitoring and these challenges show an alerting system. They are also used to highlight areas by careful planning and execution, a dependable and efficient way may be ensured for deploying applications on a Kubernetes cluster.

CHAPTER 3

REQUIREMENT & SPECIFICATION

3.1 Requirement Collection and Analysis

Identify Stakeholders: The main consumers of this work are developers of the system, which consists of a set of software programs. This report will discuss the various roles of the entities that engage with the CI/CD pipeline or the deployed applications including the developers, system administrators, and end-users. This includes the functional security and safety requirements priorities.

Define Project Scope: The project is designed to construct an integrated and efficient CI/CD pipeline for the organization. Includes monitoring and alerting systems for the orchestration of applications on a Kubernetes cluster. The scope entails among others ensuring high availability and scalability of the deployed applications and their security.

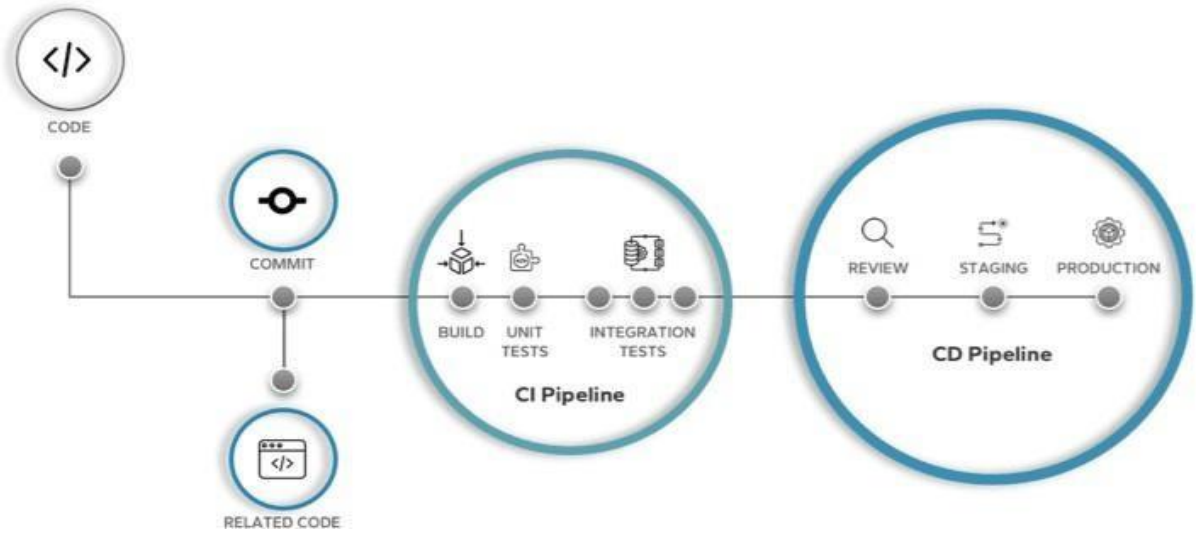
Collect Functional Requirements: The functional requirements were identified as derived from the stakeholders. A few of these are the building and testing and analyzing with SonarQube as well as the need for image scanning with Trivy. Configuration update of the Kubernetes deployment manifest and the process of automatic deployment of changes using ArgoCD. They also included installing monitoring using Node Exporter and Kube-State metrics configuration. Prometheus and Grafana as its front end. Non-functional requirements other aspects such as performance, maintainability and reliability. The factors considered earlier and based on which the objectives of the study were formulated include scalability, reliability, availability and usability. These include the need for the community.

Prioritize Requirements: Accordingly, the criteria formulated met the following objectives. The requirements were prioritized according to their influence and alignment with the phase goals and the overall goal of the project. For instance, the automation of this pipelines go efficiently a long way in solving issues. Two areas were considered essential, which are the CI/CD pipeline and the monitoring with a notification system were therefore prioritized.

Document Requirements: All the collected and analyzed requirements were documented in a comprehensive requirement specification. This document serves as a guide for the development and implementation of the project.

Iterate and Refine Requirements: The requirements were continuously reviewed and refined throughout the project lifecycle based on evolving stakeholder needs and lessons learned from ongoing development and testing activities.

3.2 Use Case Modelling



3.2 Use Case Modeling

3.3 Design Requirements

The CI/CD pipeline architecture should be pipeline-based. Each stage (building, testing, SonarQube analysis, Trivy image scanning, Kubernetes deployment manifest file update, and Argo-CD automatic deployment) should be self-executing and scalable.

Real-Time Monitoring: The system must provide real-time monitoring. This should be accomplished using Prometheus and Grafana which provide metrics for each sort of data (CPU utilization, memory consumption, network I/O, disk I/O and so on).

Kubernetes Integration: The CI/CD pipeline must be integrated with a Kubernetes cluster. Any changes in the pipeline should result in the updates affecting the Kubernetes deployment manifests.

Data Processing: To meet the requirements of real time data & solutions that the system offers should be availed at real time. The SonarQube scanner analysis and running Trivy on the image modifying the Kubernetes pod templates of applications in the form of deployment manifest files.

Alerting System: For the system to function effectively it must need to include a function that would also enable the system to notify users based on some conditions. This initiates send out of notices for successful deployment and unsuccessful deployment, resource utilization that will reflect abnormalities that are associated with the Kubernetes system on the Kubernetes cluster. It should be able to give an analysis that builds as the process goes on and increase operational efficiency. The process of collecting and analyzing data for the purpose of monitoring should identify the metrics that have to be collected. The metrics to be collected should include metrics from the CI/CD pipeline and the queried data integrated with the Kubernetes cluster and leading to generation of statistical reports.

Data Collection and Visualization: This means that it must have all the data it collects delivered to Prometheus for some specific service. The result of this is stored back in the data store storage and monitoring. It is in this step that a relationship between Grafana and Prometheus which will provide a visual display of the whole system's insights into the data.

Security: The system should ensure that applications are delivered securely and that sensitive data is protected both in transit and while stored. This involves including strong security features like Trivy's secure communication routes, access limits and image scanning capabilities.

3.4 Tools Descriptions

Ubuntu Server 22.04: Ubuntu Server 22.04 is a linux distribution built on debian that is available for free and open source. Thanks to its extensive software packages, security features and long-term support. It is well-known for its dependability and is frequently utilized in server and cloud environments. Additionally, it offers five years of default automated updates for the Ubuntu main repository.

Kernel-based Virtual Machine (KVM): KVM is a complete virtualization solution with virtualization extensions for linux running on x86 hardware. With its help, you may transform linux into a hypervisor that lets a host computer operate several separate virtual environments. Commonly referred to as guests or virtual machines. Since KVM is integrated into the current linux code, it gains instant access to all new features, updates and advancements.

Jenkins: It is parts of the software development process can be automated by developers using this open-source automation server. It is employed in your project to automate the CI/CD pipeline's building, testing and deployment phases. The command-line program that Jenkins uses to deliver this functionality serves as the user interface for all of Jenkins' features.

Docker Hub: Docker is a cloud-based hosting service where docker users and partners can build and store profile is essentially capable of creating, deploying, storing and releasing docker container images. It offers a one-stop-shop that helps put together the characters, environments, properties and other details. In this project, docker hub use as the main registry tool. Which was used to store docker images of the application that you have deployed. When an image is uploaded in the docker hub, it they should be pulled from there and then taken to which environment and ensure that they are efficient and reliable. Its ability to act as cache for public or private images, hooks for web and handling of automatic build, which make it a crucial component of CI/CD process that helps in developing high quality software products. So as to make your deployed software handy and convenient. It ensures that you have made the necessary enhancements. Managing to run the latest and most secure versions of your images in a docker container. It’s a critical component in the CI/CD process which helps to guarantee that the applications are using the most current and appropriate version to remain secured and avoid falling for dangerous hacker’s tricks.

Docker: It is a platform that isolates applications into containers and allows developers to streamline the process of deployment, scaling and management. It uses docker in the project for building containers for your software applications. This is an assurance that compatibility of these environments will be achieved.

Kubernetes: An open-source platform that is made to simplify the deployment, scaling and management of application containers. Kubernetes controls the distribution of your docker containers among several nodes.

ArgoCD: For Kubernetes, ArgoCD is a declarative GitOps continuous delivery solution. It automatically deploys any changes to my Kubernetes cluster while keeping an eye on Git repository for updates. ArgoCD notifies the team that there are problems to be fixed using its user interface.

Prometheus is an open-source toolkit for alerting and monitoring systems. It gathers, stores and offers a robust query language for analyzing the metrics from your applications. The primary characteristics of Prometheus include a multi-dimensional data model that uses key-value pairs and metric names to identify time series data.

Grafana: Grafana is an open-source platform designed for observability and monitoring. It lets you see the metrics that Prometheus has gathered on a dashboard. Grafana allows you to create stunning, customizable dashboards.

SonarQube: SonarQube is an open-source tool for ongoing code quality inspection. It is a free code quality inspection tool that helps consistently monitor the quality of the Codebase, to find errors, compile time errors, syntax errors and security issues. It then checks the code to people using static analysis technique, code redundancy, coding constraints, test effectiveness, code density, documentation and errors. The report of SonarQube has important and informative data that can be summarized as code smells, technical debt, reliability and security.

Trivy: It is one-stop open-source vulnerability scanner that helps analyze containers and others that have been derived from the use of pool artifacts. It is a basic tool that can be highly dependable especially when it is scanning for the vulnerabilities in the system. Container images, file systems. Trivy is used in continuous integration and delivery process to check your application's docker images for any detected weakness. This ensures that your deployed applications are robust, secure,

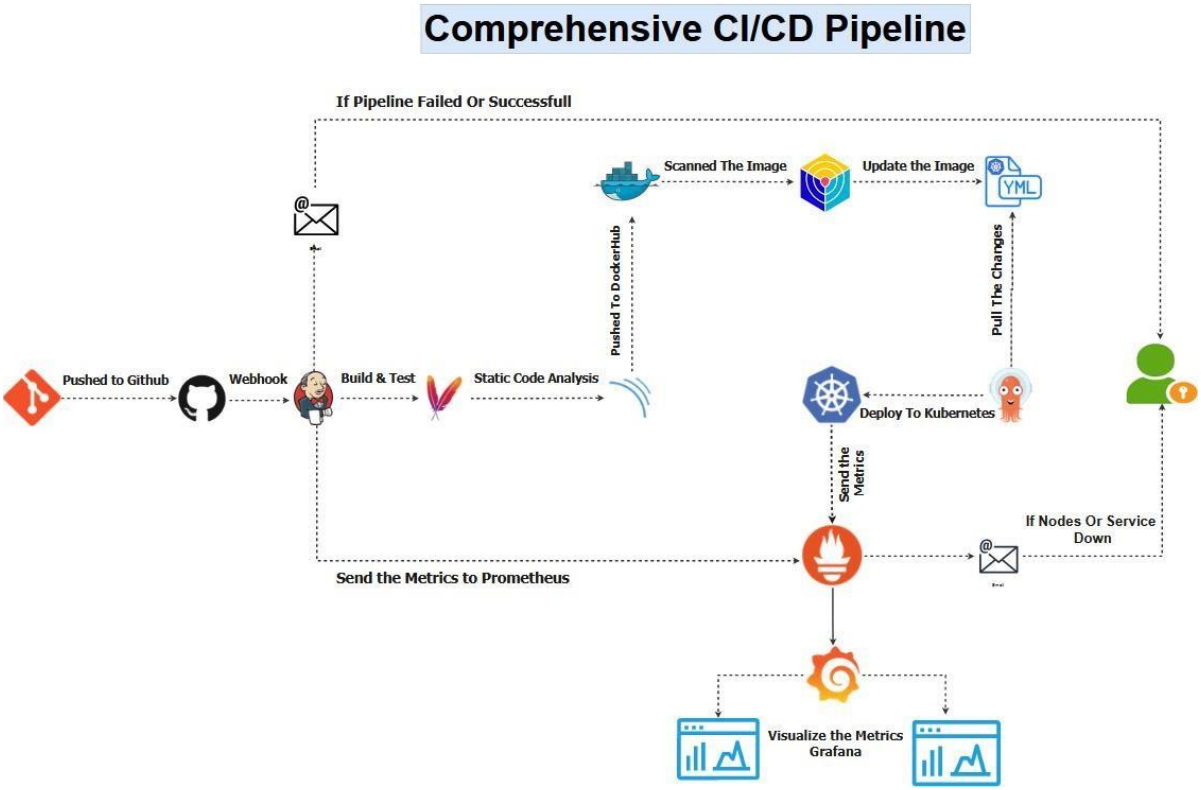
and reliable for the provision of services. The advantages of Trivy include reliability, high speed, and convenient integrated operations. Which makes it one of the most popular facilities in security scanning in CI/CD pipelines.

Altert-Manager: Alert-Manager is a complex that is the part of Prometheus stack and it focuses on handling alerts that is produced by Prometheus monitoring and alerting solution.
It is responsible for managing and delivering alerts and ensuring that the right people receive information about incidents.

CHAPTER 4

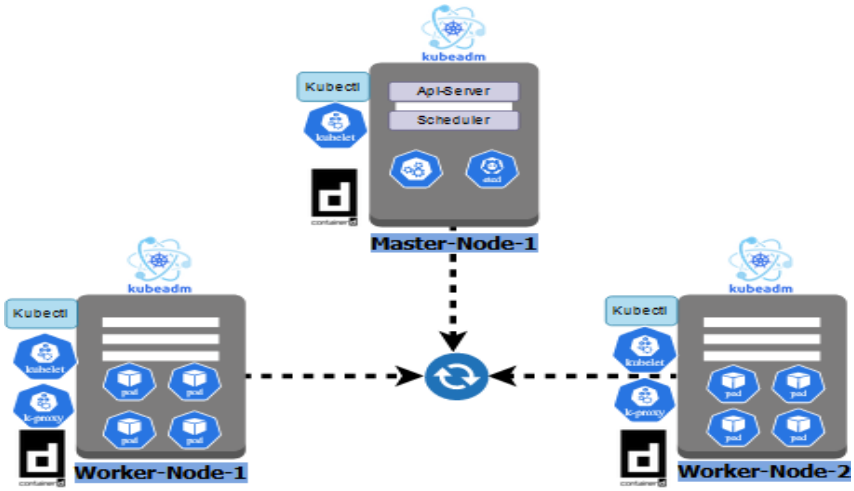
DESIGN SPECIFICATIONS

4.1 Architectural Design



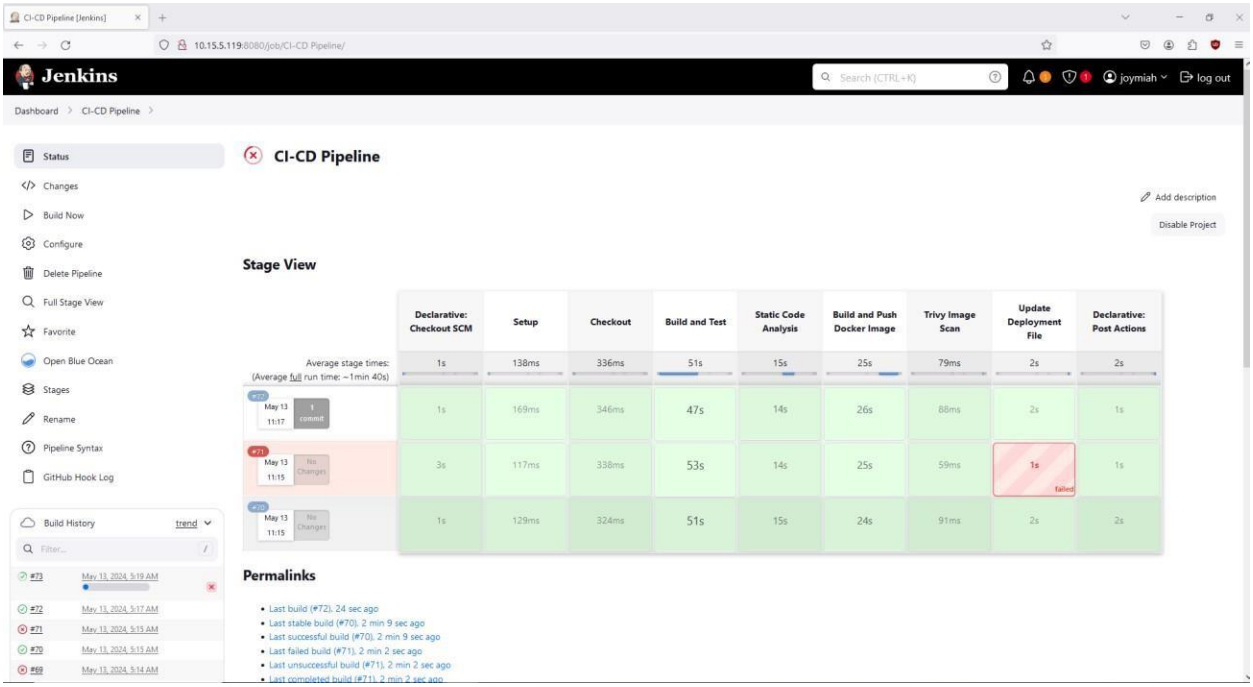
4.1 Architectural Design

4.2 Kubernetes Cluster Architectural Design



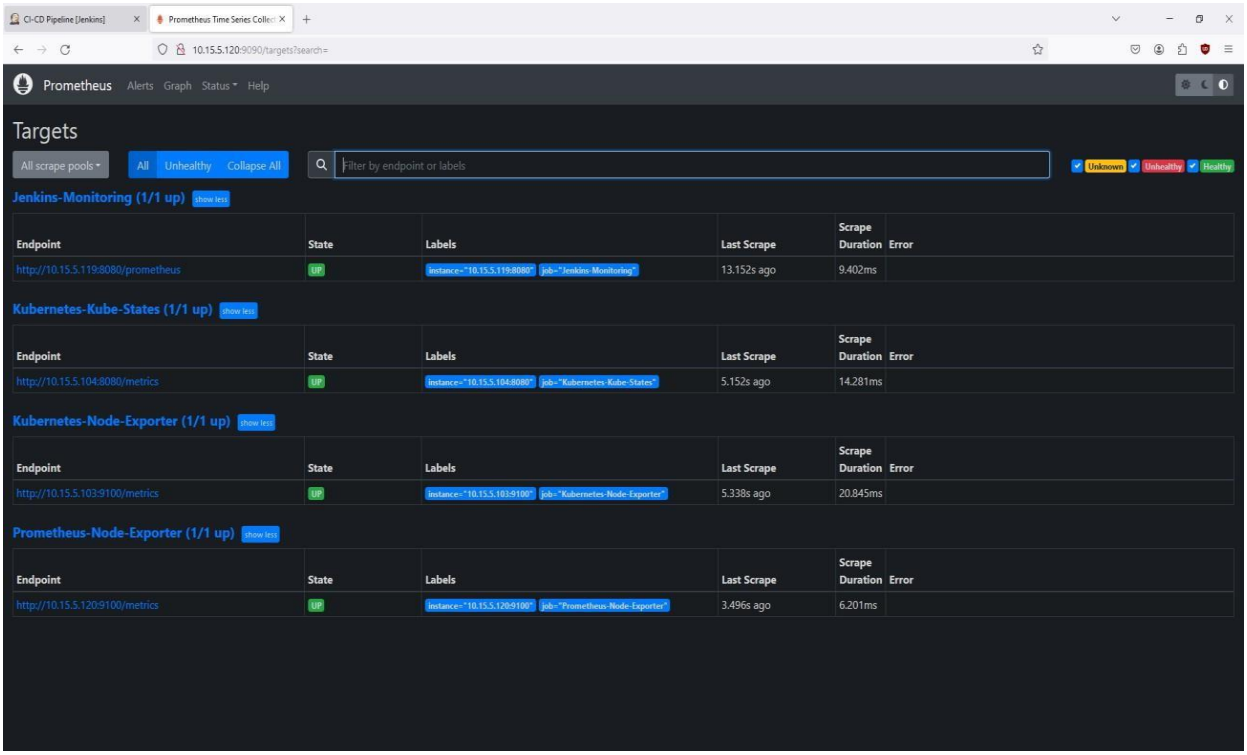
4.2 Kubernetes Cluster

4.3 CI/CD Pipeline Stage Design



4.3 CI/CD Pipeline Stage Design

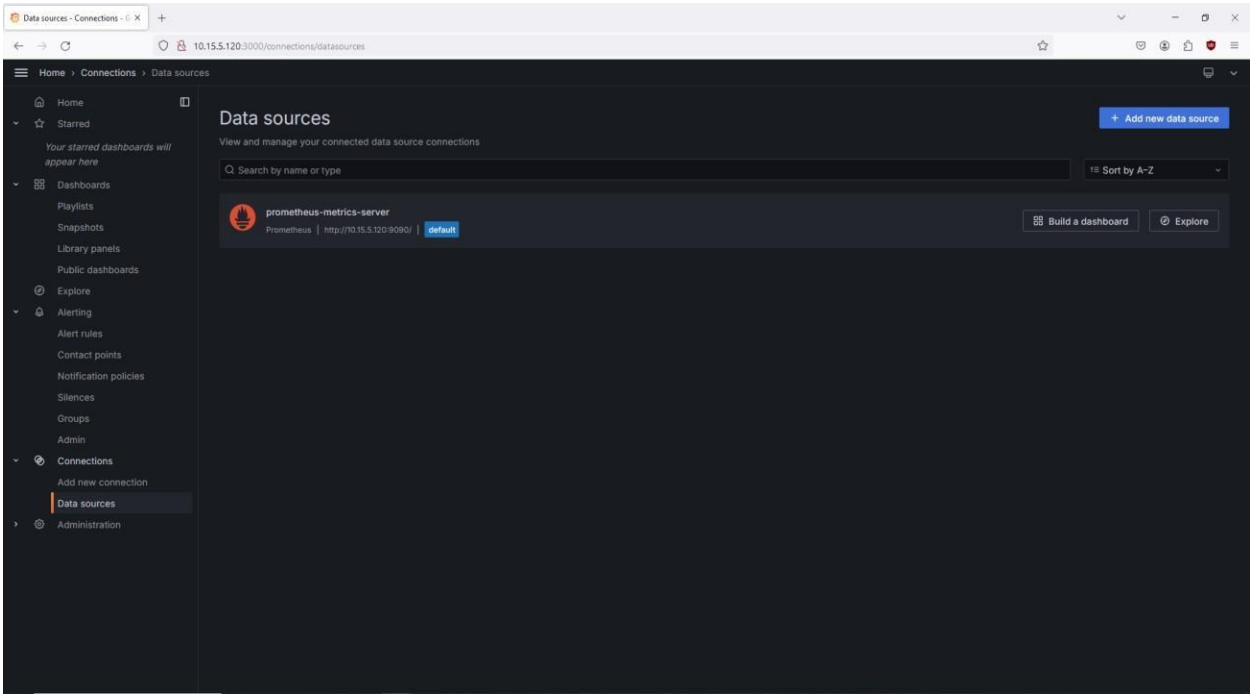
4.4 Monitoring Targets



4.4 Monitoring Targets

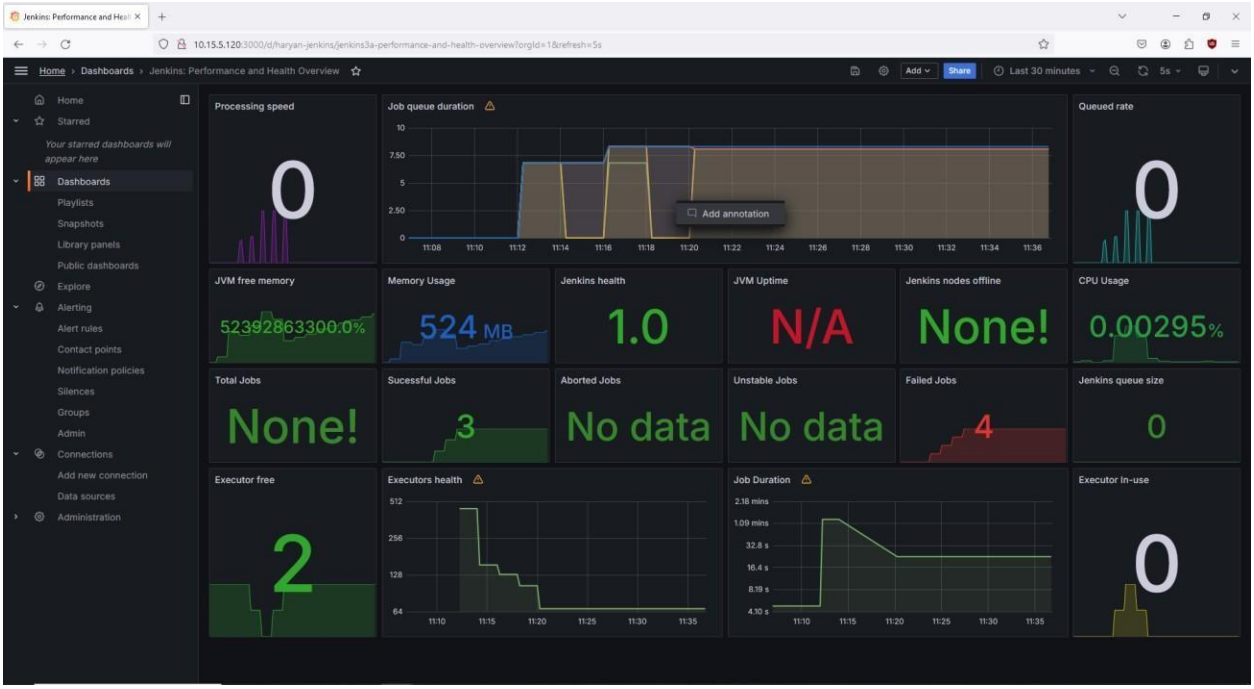
4.5 Grafana Dashboard Designs

Grafana Can Contains Multiple data sources and Multiples Dashboards:



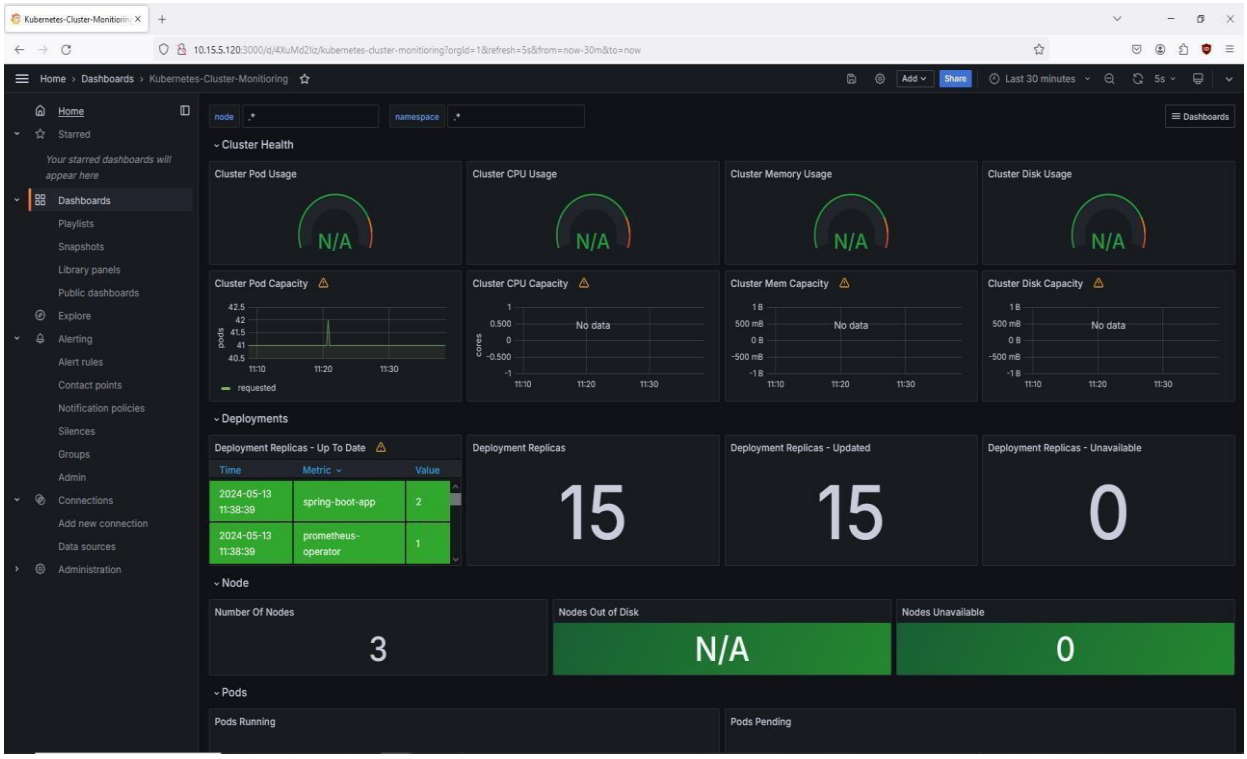
4.5.1 Data-Source

Dashboard-1: Monitoring Jenkins Pipeline



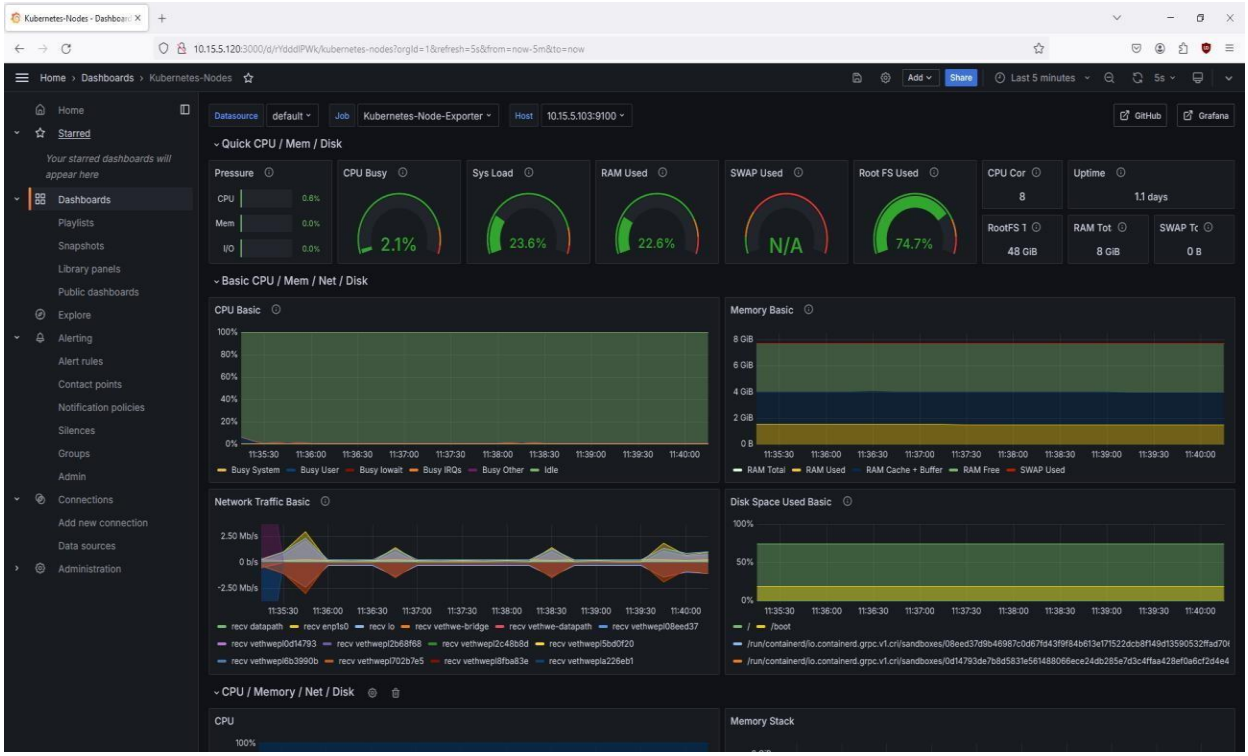
4.5.2 Jenkins Pipeline Monitoring

Dashboard-2: Monitoring Kubernetes Cluster Components



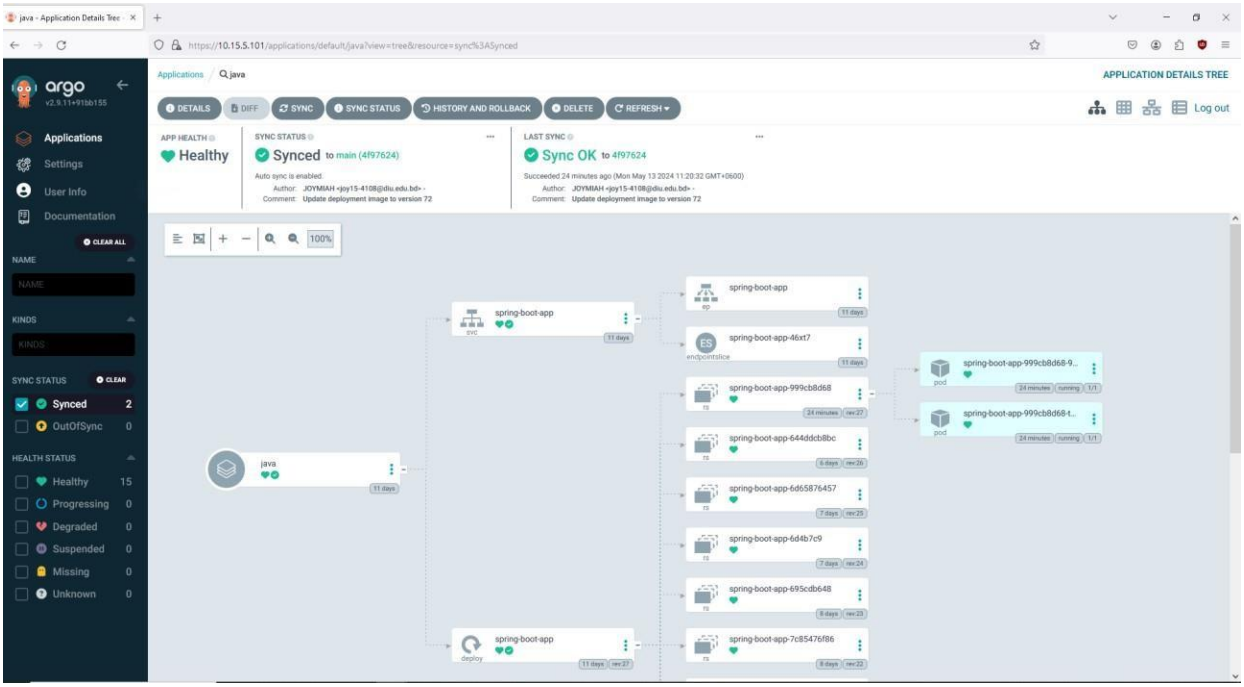
4.5.3 Kubernetes Kube-State Monitoring

Dashboard-3: Kubernetes Nodes



4.5.4 Kubernetes Nodes Utilizations Monitoring

4.6 Argo-CD Design



4.6 Argo-CD Interface

4.7 Implementation Requirements

This project one of the major goals is the creation of a CI/CD pipeline. That aims to be set up in the most optimal way. It should be designed to run naturally always, whenever a new file has been pushed to the GitHub repository. The pipeline should contain the stages of developing, testing, SonarQube analysis, scanning of images through Trivy and updating the images to the Kubernetes deployment manifests.

Kubernetes Cluster Configuration: Kubernetes cluster should have one master node and multiple worker nodes in the cluster. Thus, the cluster should be configured to pull the changes from the GitHub repository. The deployment can done through ArgoCD which is an open-source tool for the deployments of the application.

Monitoring System Setup: Improve the usability of Prometheus and Grafana for the task of real time monitoring of applications. The CI/CD pipeline and kubernetes cluster building blocks are introduced as the largest in size and usage in this particular project. Other parameters that can be used are the usage of CPU, memory among others for a particular application.

Alerting System Configuration: The initial alerting system must be implemented to send email alert depending on particular criteria. The above alerting system should be used paralleled monitoring system in the above proposed alerting system. It is used in paralleled monitoring systems and cases like the successful and or failed deployment and usage of resources either in high or low ratios. The evaluation should include the effectiveness and efficiency of the application.

Scalability and High Availability: As suggested high availability should be implemented on the Kubernetes. This means it is possible to install several worker nodes that provide the perfect balance as well as input-stream enabling automation based on the extensibility of the resources and features of the system used.

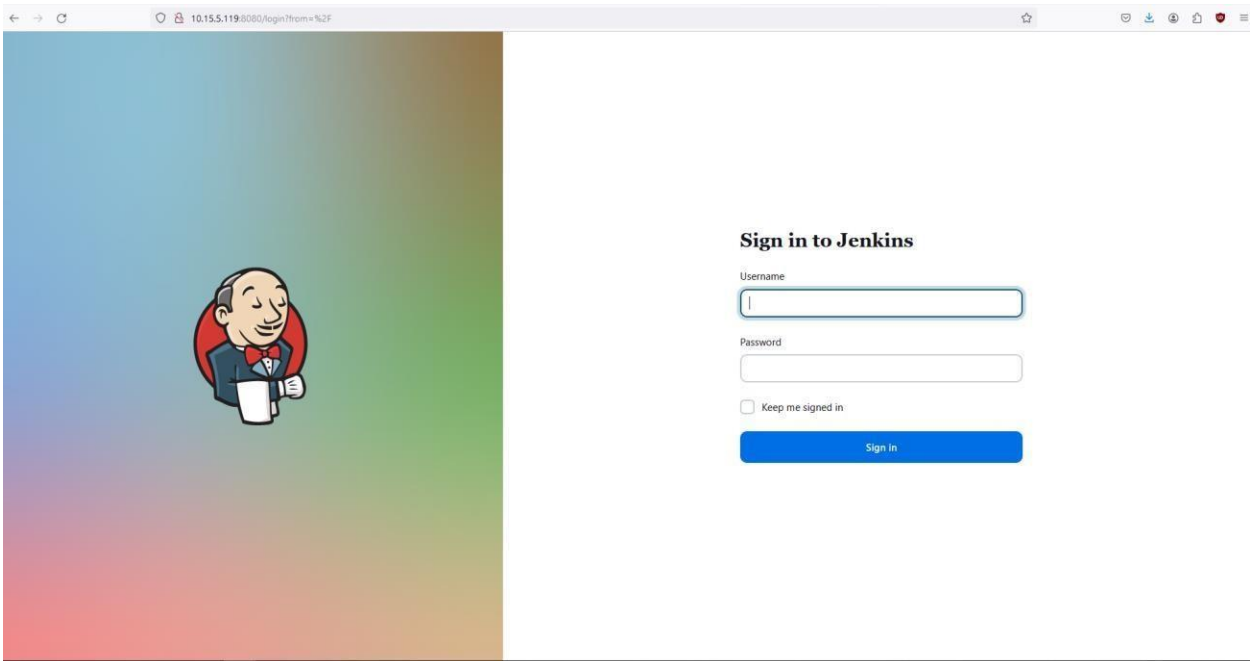
CHAPTER 5

IMPLEMENTATION AND TESTING

5.1 Implementation of CI/CD Pipeline Stages

The implementation of the CI/CD pipeline stages:

Set-Up Jenkins:



5.1.1 Jenkins Landing Page

Building Stage: This stage featured the construction of the application using this build tool which had been predetermined. The build included in the software build process was the process of creating the source code, the running of test units, and the making of an executable file or an object file. A docker image of the application to be able to successfully run the container that contains the application.

Testing Stage: After the build stage, the efficiency of the application to determine if it is fully operational was expected. This included integration tests, functional tests, as well as performance tests. In a functional testing phase, any problems that were discovered then the problem must be resolved before proceeding to the next test.

SonarQube Analysis Stage: In this stage, the SonarQube tool was used to analyze the code and identify different kinds of problems. This enabled one to find the presence of code smells, bug, or even security flaws in the codes. The results of the SonarQube analysis were carefully discussed and in case if there were some essential problems they were solved.

Image Scanning with Trivy Stage: After SonarQube analysis the docker image of the application was analyzed using the Trivy which is a container image vulnerability scanner. This helped in determining any security risks in the image or file system.

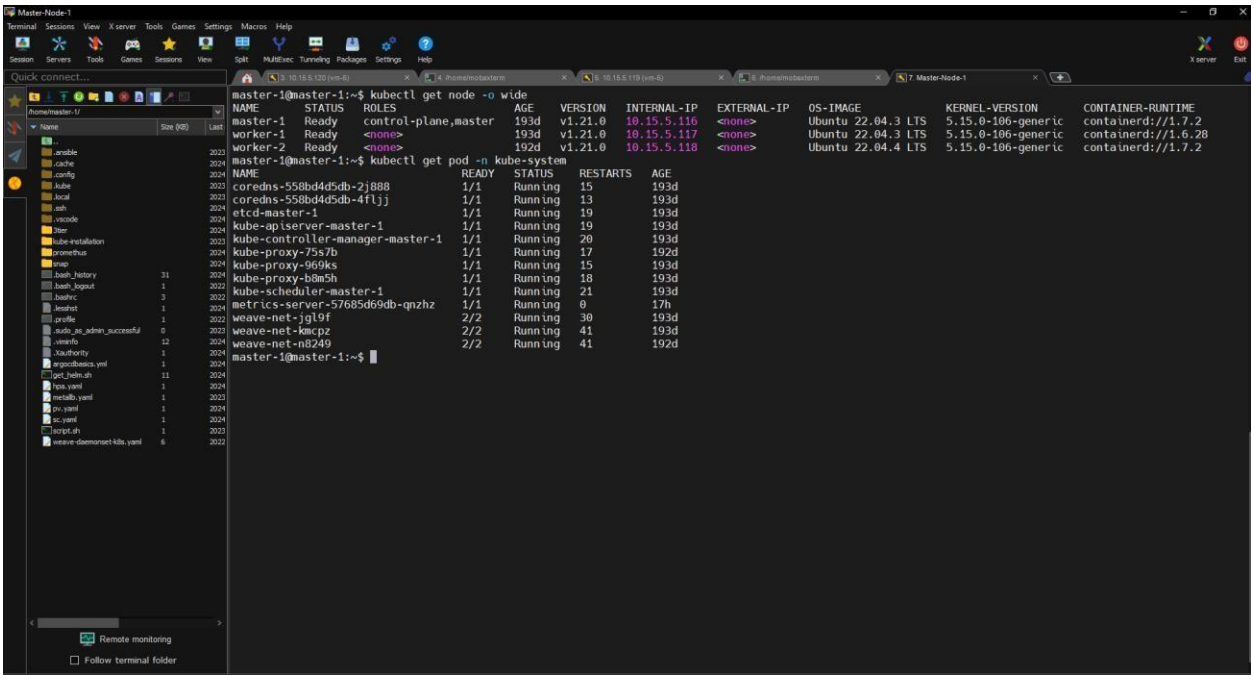
Updating Kubernetes Deployment Manifest Files Stage: After SonarQube analysis the docker image of the application was analyzed using the Trivy. Next update the new image tag to the deployment manifest file of Kubernetes.

Automatic Deployment using Argo-CD Stage: After the Kubernetes deployment manifest files are updated, the old configurations must be removed from the Kubernetes cluster. ArgoCD constantly looked at the GitHub repository to check for any updates concerning the configuration and automatically updated across the Kubernetes cluster. All of these stages were carried out in the Jenkins pipeline that was created to build the docker image. The pipeline was configured to monitor and call after a push to the GitHub repository by an automatically prescheduled event. The deployment process in as more effective and less prone to problems as possible.

5.2 Implementation of Kubernetes Cluster with ArgoCD

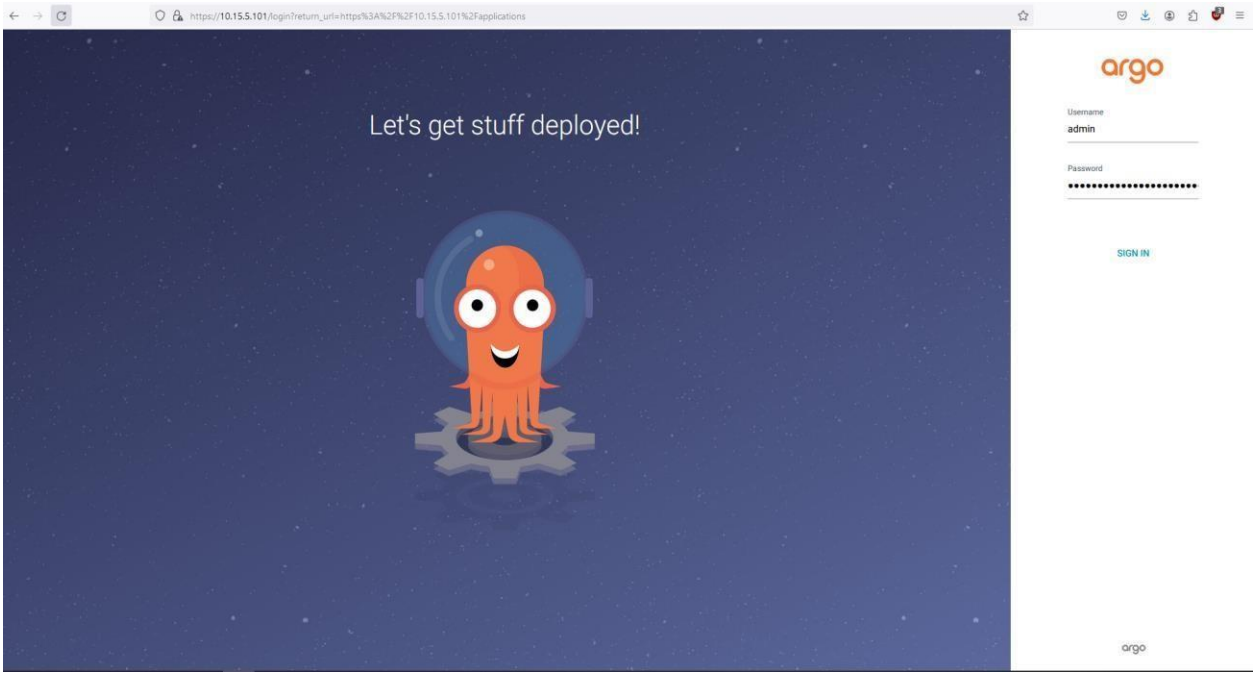
The implementation of the Kubernetes cluster with ArgoCD was carried out as follows:

Kubernetes Cluster Setup: I'm going to setup a cluster that contains one master node and two worker nodes. There was the master node, the primary node that represented the cluster and the worker nodes used to run the Application. This was done with aim to make the cluster highly available and also can easily scale up of the applications.



5.2.1 Kubernetes Nodes & Kube-System Pods

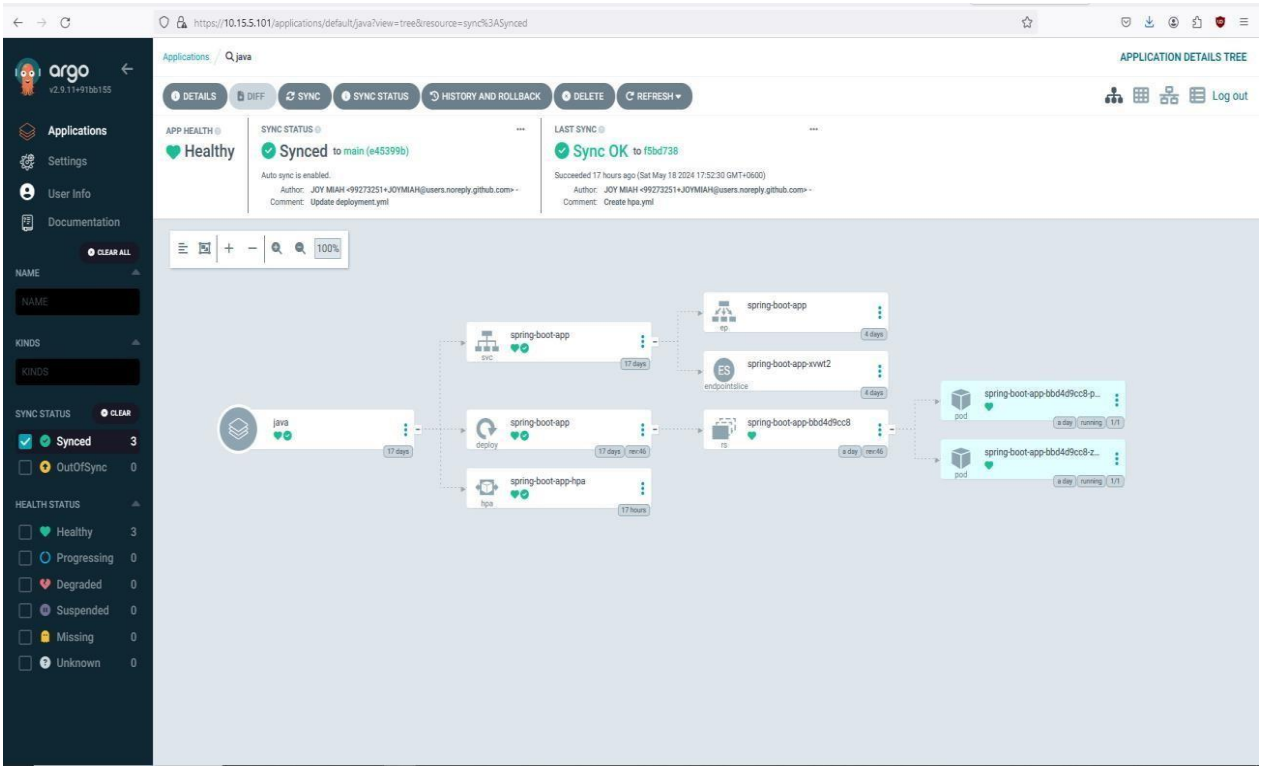
ArgoCD Installation: ArgoCD which is a gitOps declarative tool for continuous delivery for Kubernetes. It deployed in the Kubernetes cluster as operator & exposed service as Load-Balancer types. It was selected due to its ability to continuously sync with the git hub repository of a resource and apply any changes make changes to be rolled out to the Kubernetes cluster.



5.2.2 ArgoCD Login Page

Application Deployment: Through Argo, the applications were deployed on the Kubernetes cluster as follows. It was set to use the ArgoCD to track changes in the GitHub repository and apply the changes in the deployment to the Kubernetes cluster.

Monitoring and Alerting: ArgoCD was also configured for real-time monitoring as well as feedback that is when a user makes a request to deploy an application in the cluster it responds with a statement. This included ensuring that they kept an eye on the status of the deployment.



5.2.3 ArgoCD Monitoring

Security Measures: The Kubernetes cluster was surrounded with great security concerns to enhance its security level to the deployed applications. This comprised issues such as developing secure communication systems and the adoption of secure networking methods by putting certain levels and rights of access controls. The goal of the deployment process is made clear and it's efficiency enhanced and ensuring a reliable process. It also ensured that the applications was able to execute its tasks a lot better.

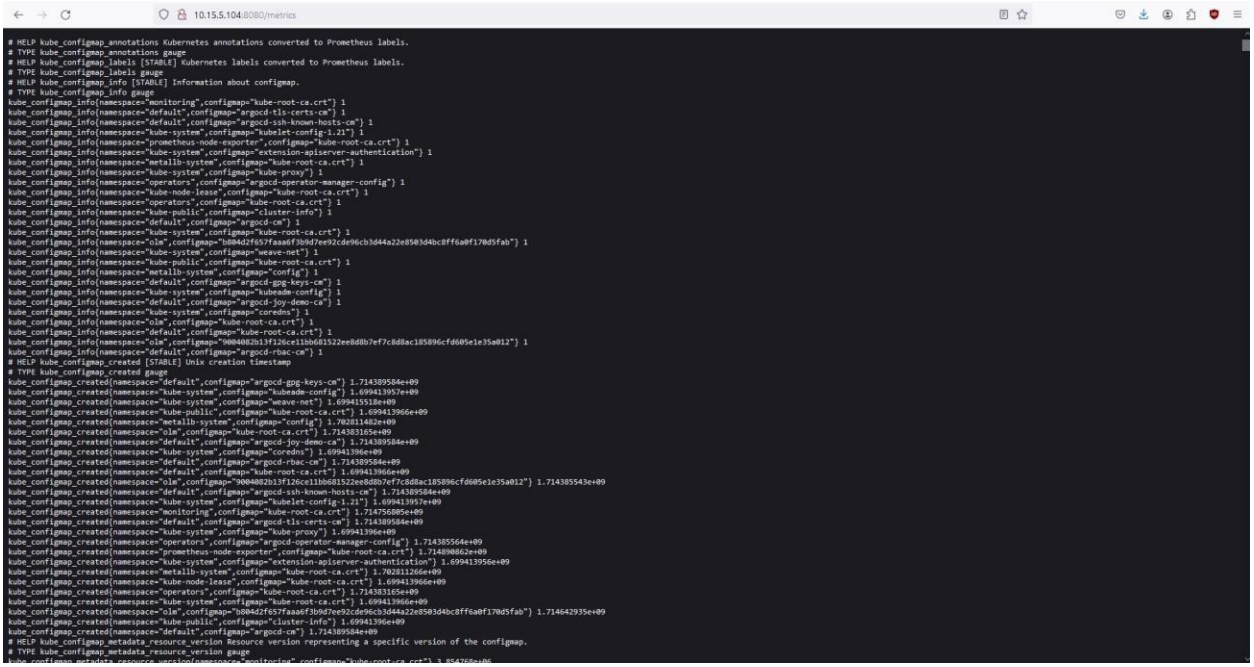
5.3 Implementation of Prometheus

Here is the Prometheus monitoring system configuration:

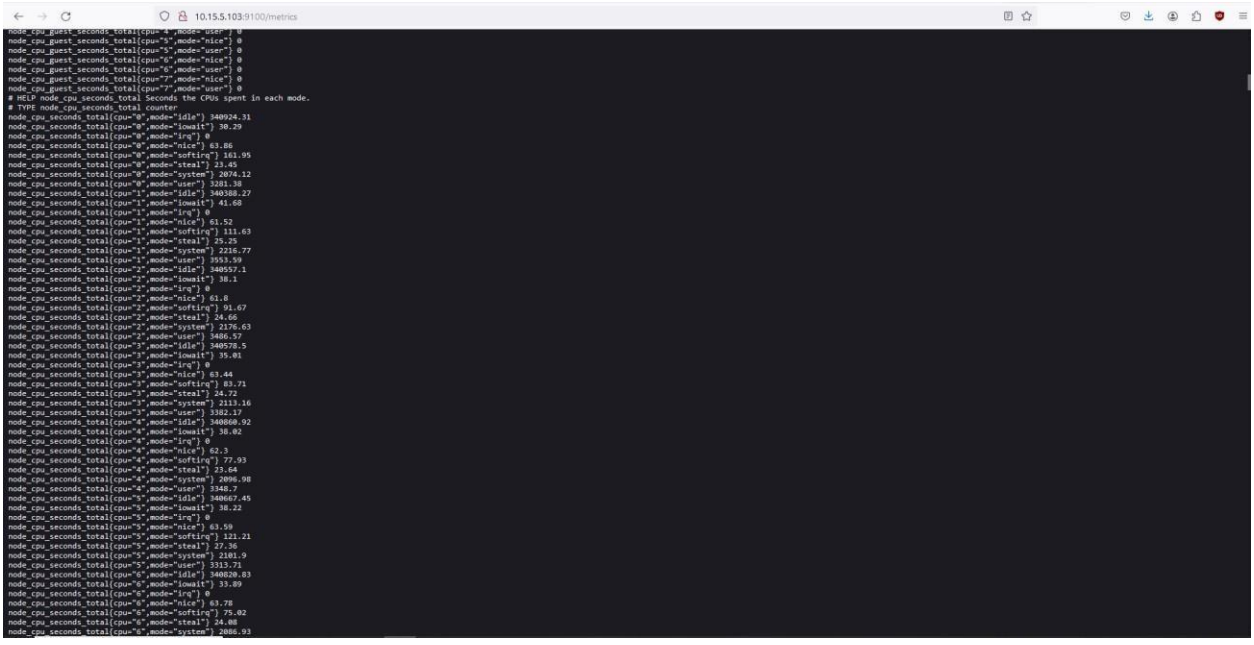
Prometheus Setup: Open source tool designed to monitor operational systems and issue alerts when abnormal conditions are identified intervals during the CI/CD process and the Kubernetes Cluster. Node-Exporter and Kube-State metrics are installed into all Kubernetes nodes, which gives nodes metrics and Kubernetes object state. Indeed Prometheus hides and stored all the data for further analysis and visualization.

5.3.1 Prometheus Landing Page

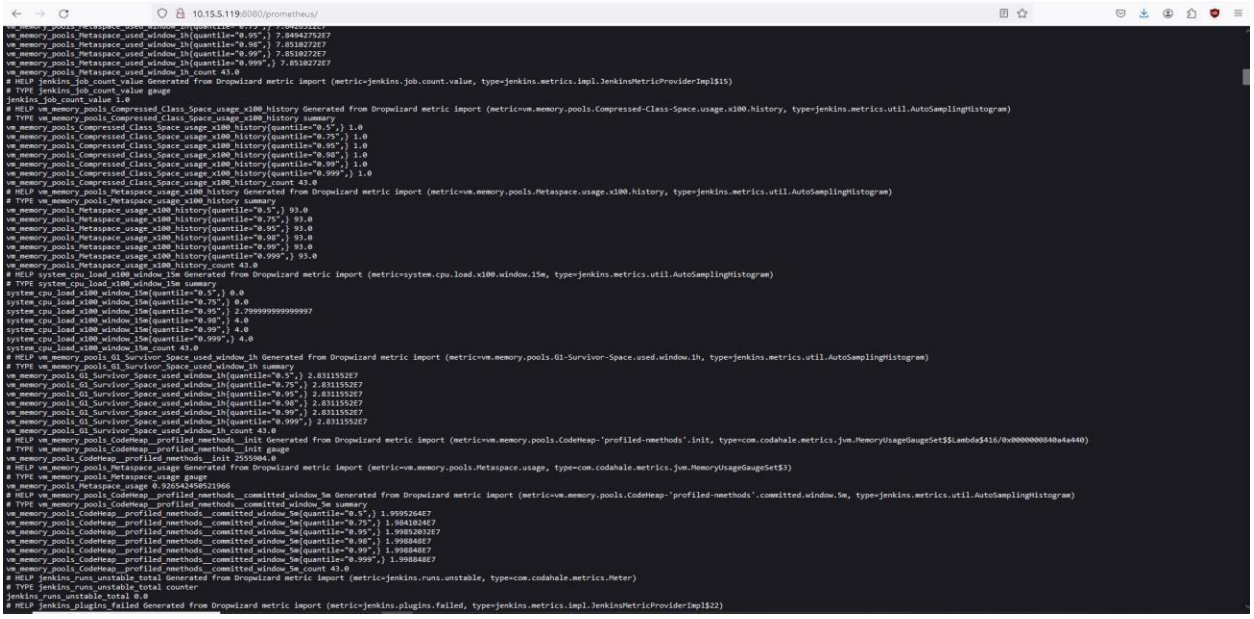
Metrics Collection: Prometheus initially assumed a high load task. It was designed to gather as much data as possible from targets like CPU, snap, memory, net and disk teams, procedure, etc. These metrics gave useful insights and describing the performance of CI/CD pipeline and the Kubernetes cluster.



5.3.2 Kubernetes-Kube-State-Metrics

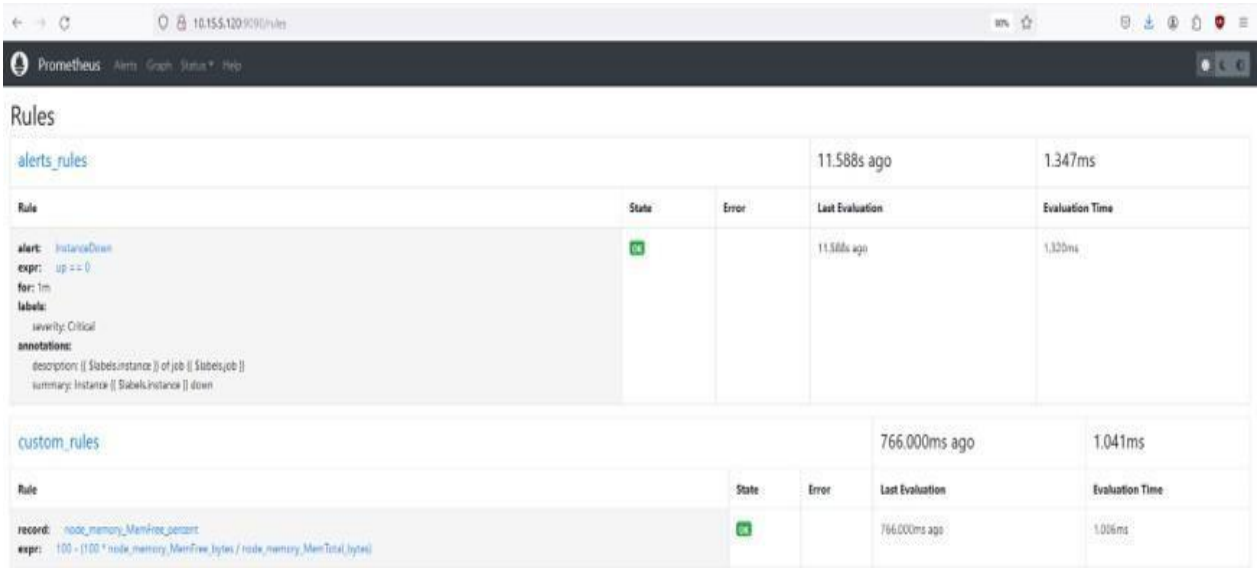


5.3.3 Kubernetes Node-Exporter Metrics



5.3.4 Jenkins Metrics

Alerting Rules: Prometheus has the feature of alerting rules, which alerts can be set up in the specific format of alarms in a certain criteria are satisfied. Notice tests were designed to be produced when the defined attributes were breached or the hardware was over utilized. These notifications were set to be delivered by default to an electronic mail of an alerting system that sends a notification on a particular recipient on an event.



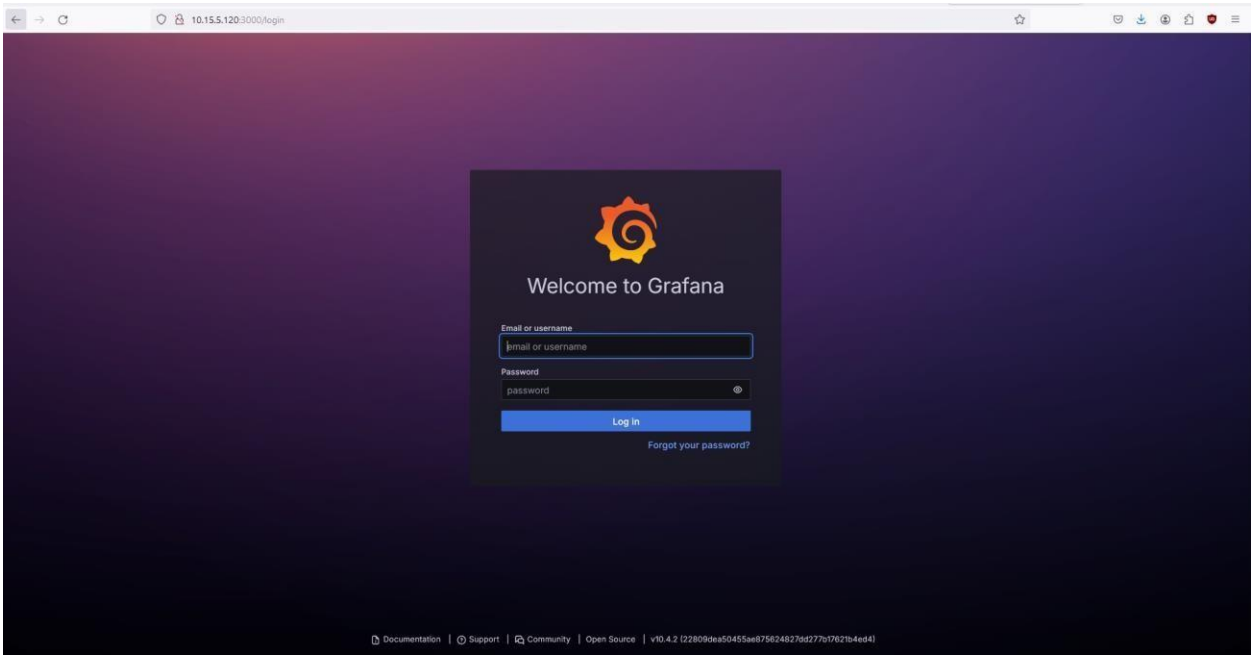
5.3.5 Alerting Rules

Integration with Grafana: Grafana is open-source large-scale real-time data processing and collaborative interactive visualization system for cross-platform environment. It connected to the Prometheus to as primary data source and visualize those data based on metics.

5.4 Implementation of Grafana

The specifics of Grafana implementation were as follows:

Grafana Setup: The Grafana, an open-source tool used primarily for monitoring and understanding systems. It was developed to work as a part of which it serves as a data source to connect to Prometheus. So that it can aggregate the metrics and introduce the data collected in front of the human lens.



5.4.1 Grafana Login Page

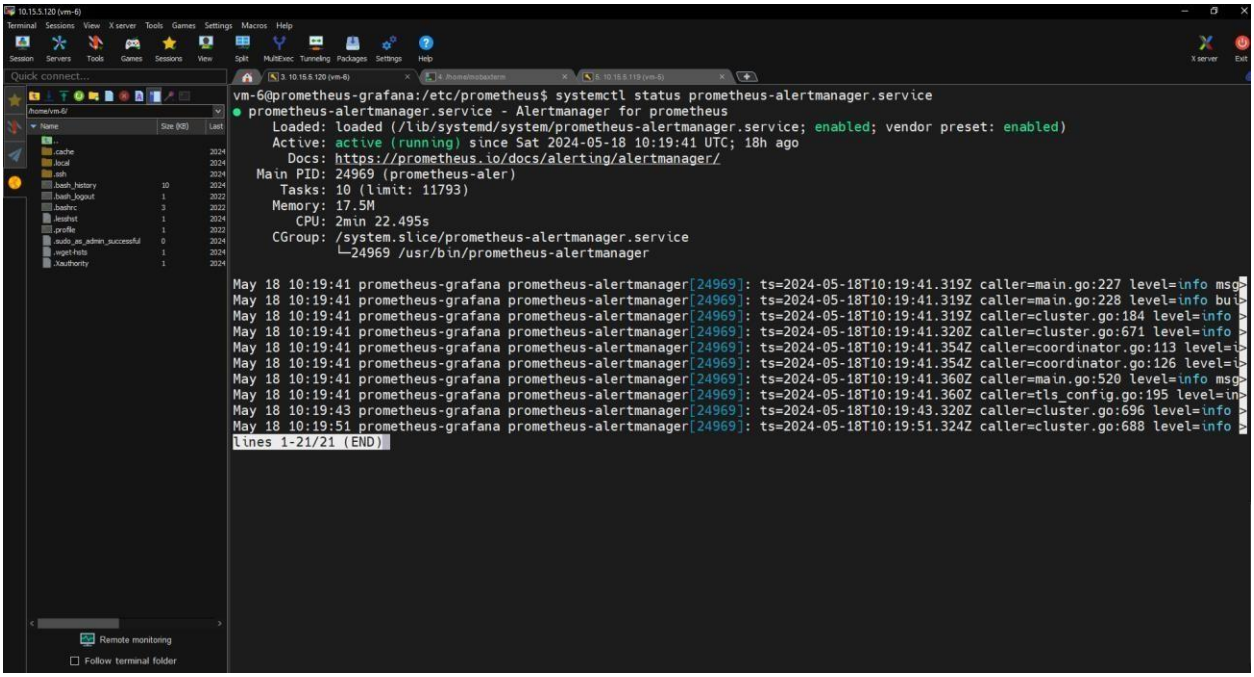
Dashboard Configuration: To achieve this, a dashboard on Grafana was created for the purpose of viewing massive data Prometheus. In short the dashboard was the primary statistics synonymous with CI/CD an adequate surveillance of the global load in the pipeline and the managing Kubernetes cluster like the CPU score, memory, network and disk, I/O. Grafana have the user friendly UI that anyone can easily understand that what is going on.

Security Measures: Grafana dashboard involved for creating secure channel of communication as well as giving access overseeing human endeavors.

5.5 Implementation of Alerting System for Email Alerts

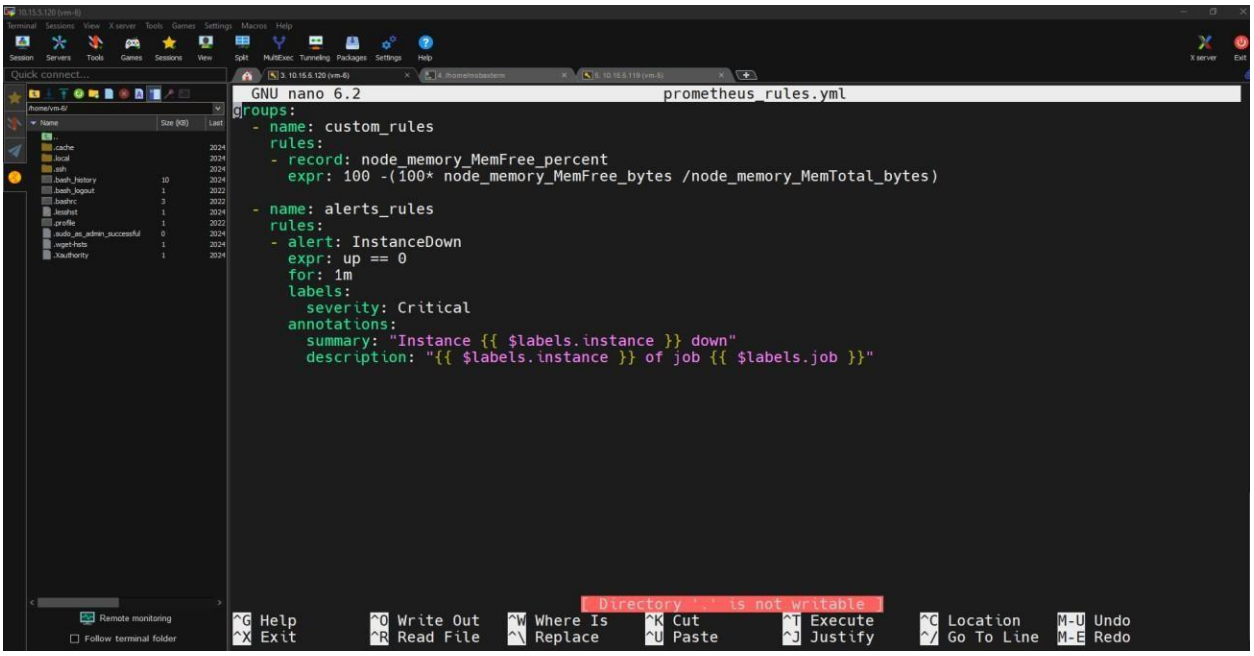
The implementation of the alerting system:

Alerting System Setup: To address this issue an alerting system that was developed to send email alert was created particular conditions. This system was integrated with Prometheus and Grafana to have end to end.



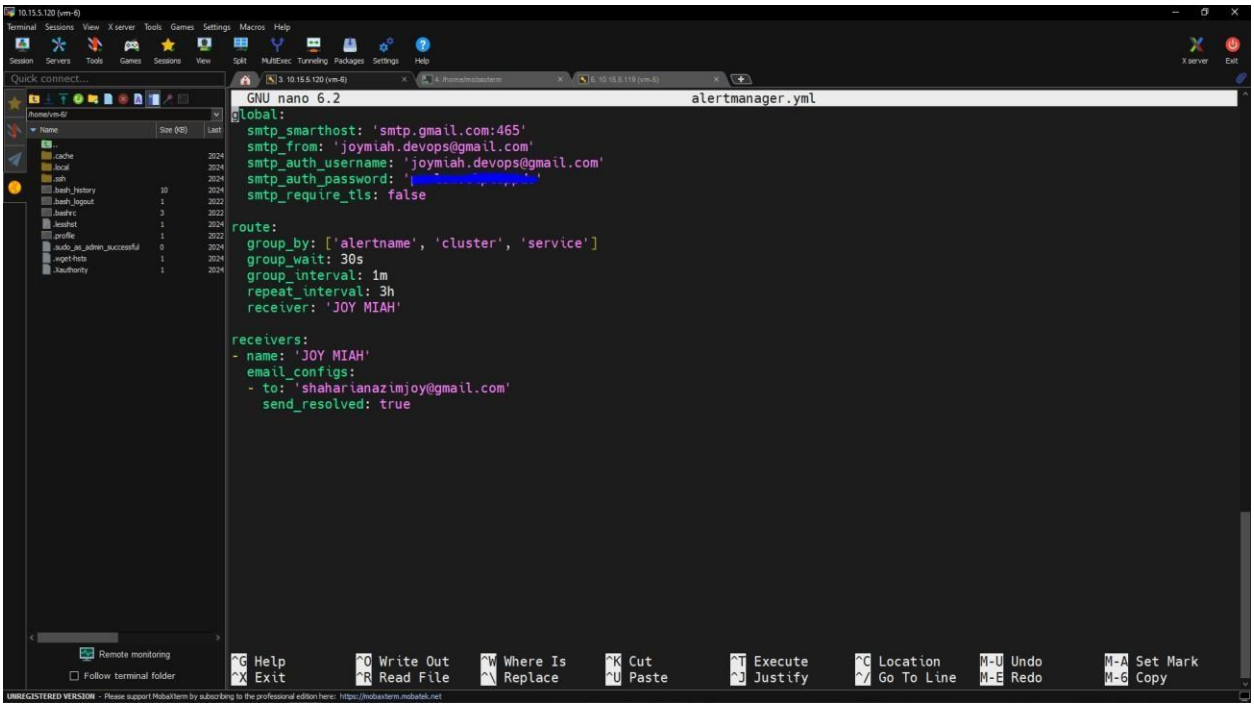
5.5.1 Set-Up Alert-Manager

Alerting Rules Configuration: As for Prometheus, it has obtained alerting rules which aimed at sending alerts depending on particular criteria. These situations include substantial resource consumption, cluster deployments and issues found in the Kubernetes environment.



5.5.2 Alerting Rules Configuration

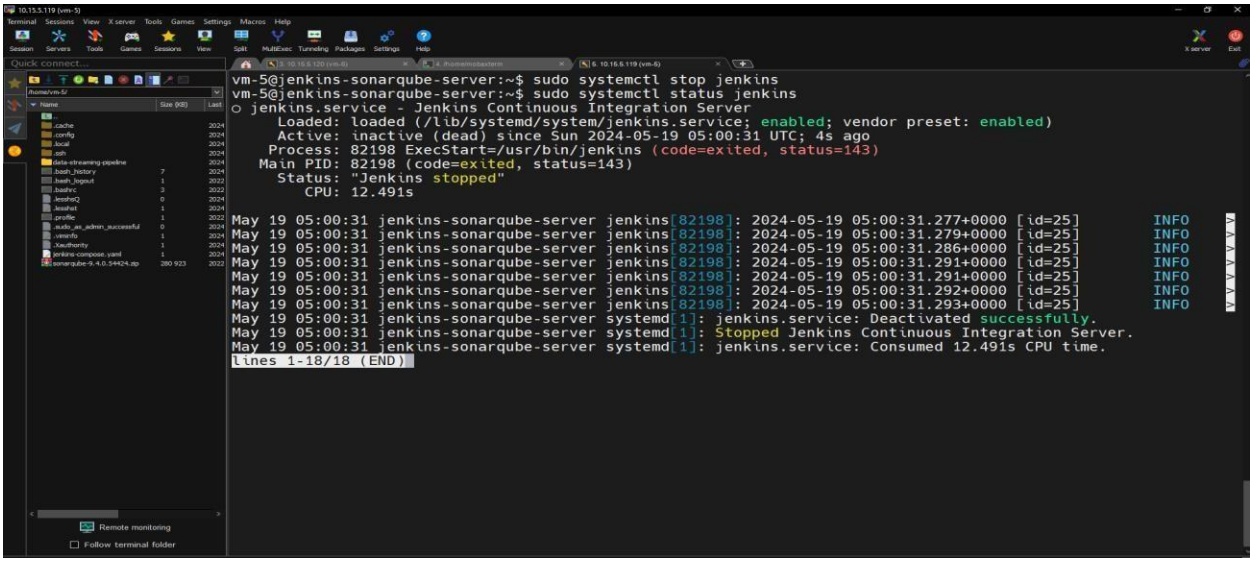
Email Alert Integration: The alerting system was integrated with an email service to send results in the form of an email alerts to the right consumer, user and investors. The email notifications they received included comprehensive provisions of detailing the warning situation include the time that incident occurred.



5.5.3 Email-Alert Configuration

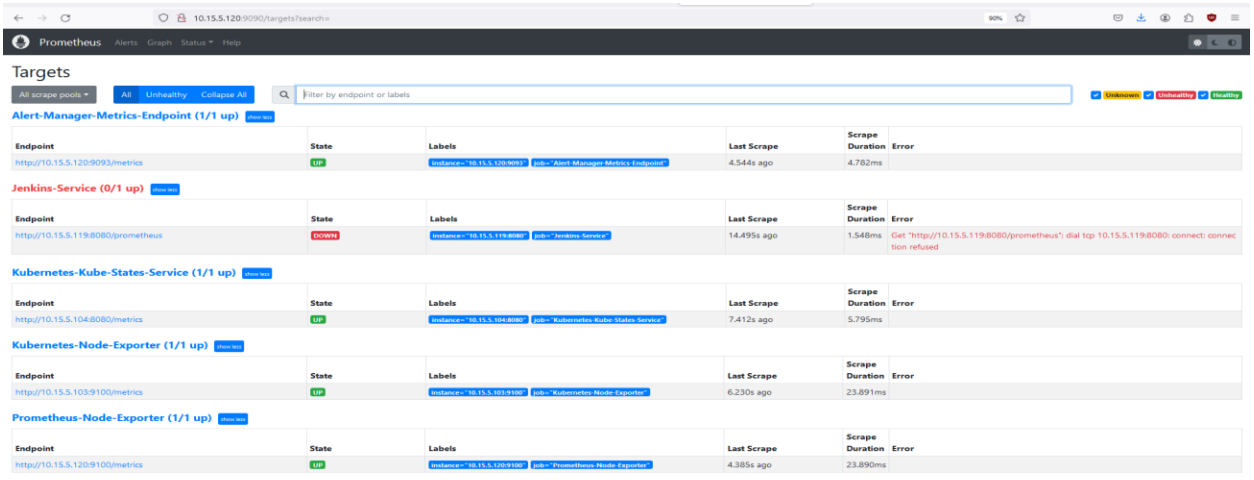
Alert Testing: The alerting system clearly having been tested to ensure that it would be useful in the proper way. This should also comprise the analysis of the alert triggering mechanism, the system for the e-mail delivery process, as well as the comprehensibility of the message. Accuracy of the data provided in the mail messages of the email notification.

1. Shutdown a Service or Instance: First, turn off one of the services or instances you have installed on your server, either to test connectivity or to simply start over are monitoring.



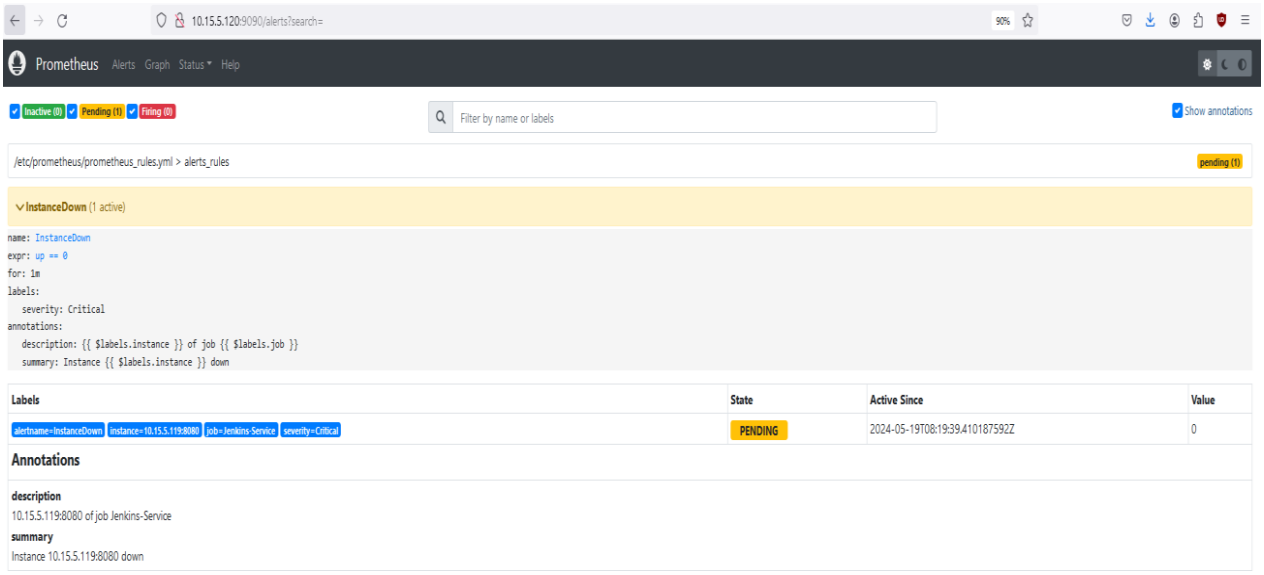
5.5.4 Shutdown Jenkins Service

2. Check Prometheus Target Interface: Now open the Prometheus target interface. You will notice that Jenkins service is down. The status can be checked in the following link.



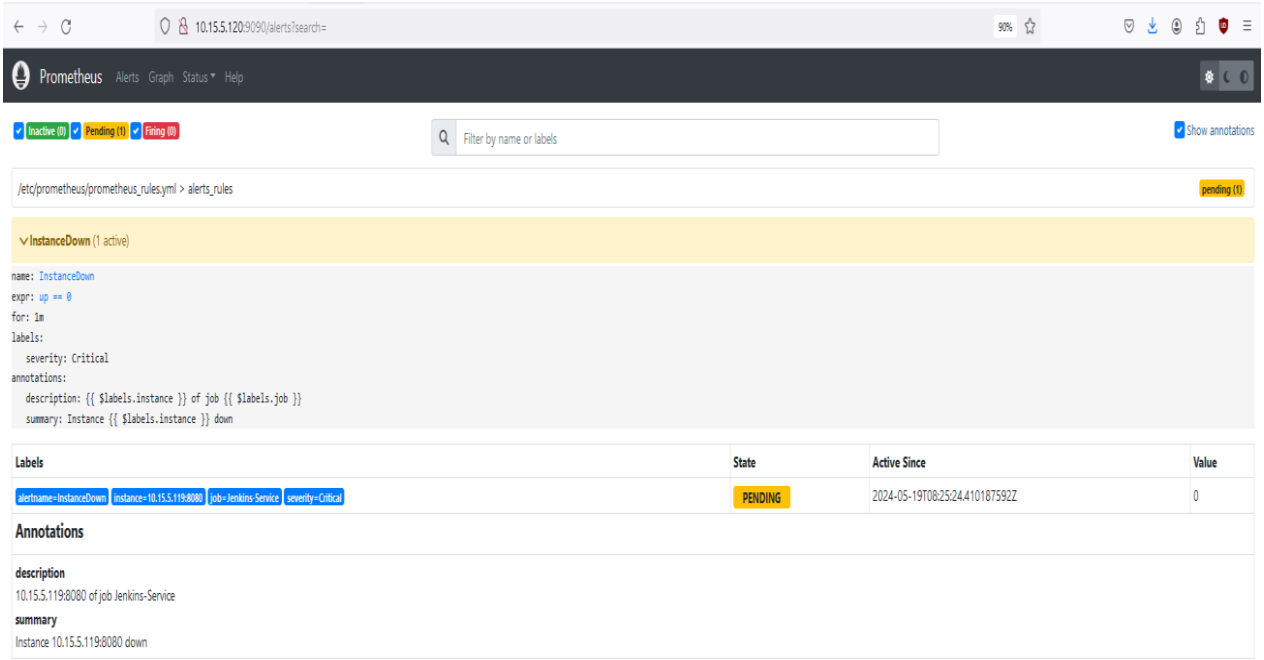
5.5.5 Check Prometheus Target

3. Observe Prometheus Alerts: There are many places in Prometheus. The one to select is called the Alerts section. Lastly, the instance is flagged said to be inactive. First, the alert will be in the pending state.



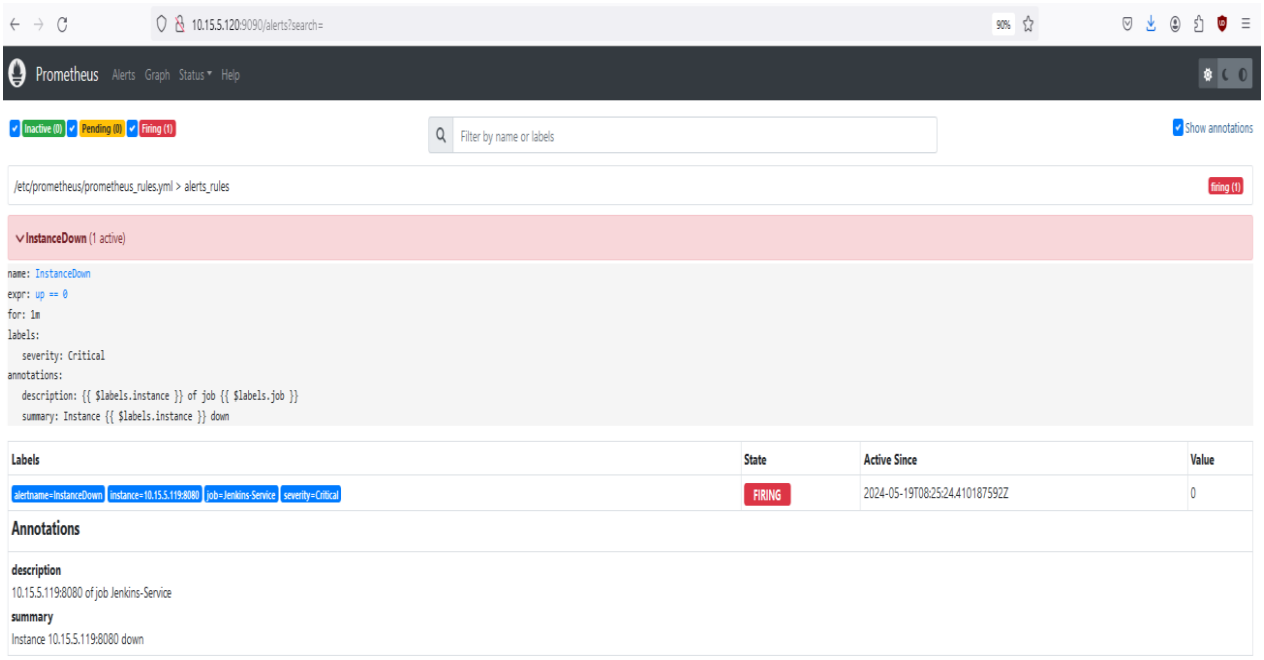
5.5.6 Check Prometheus Alerts Interface

4. Pending Stage: The alert will be activate the Pending state for one minute. If Jenkins does not startup within one minute then the alert moves to the next level which is a firing alert.

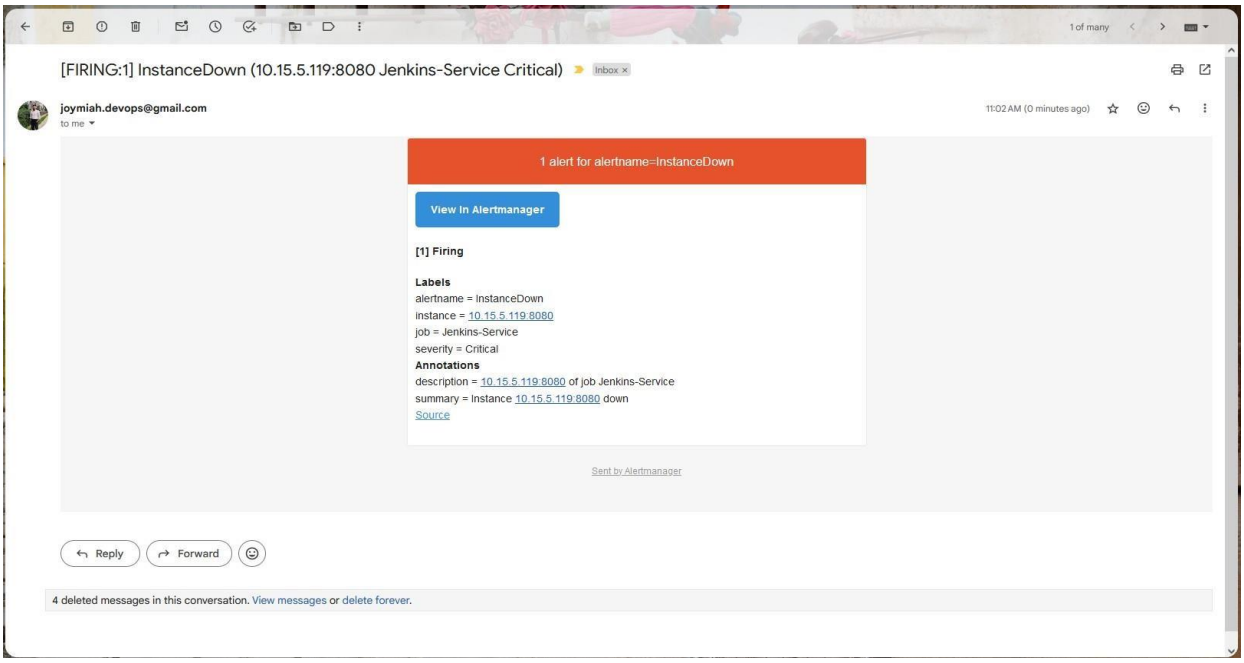


5.5.7 Check Prometheus Alerts Interface

5. Firing Stage: After it gets to the firing stage, Prometheus will send an email to those emails and tell them Jenkins service is down.

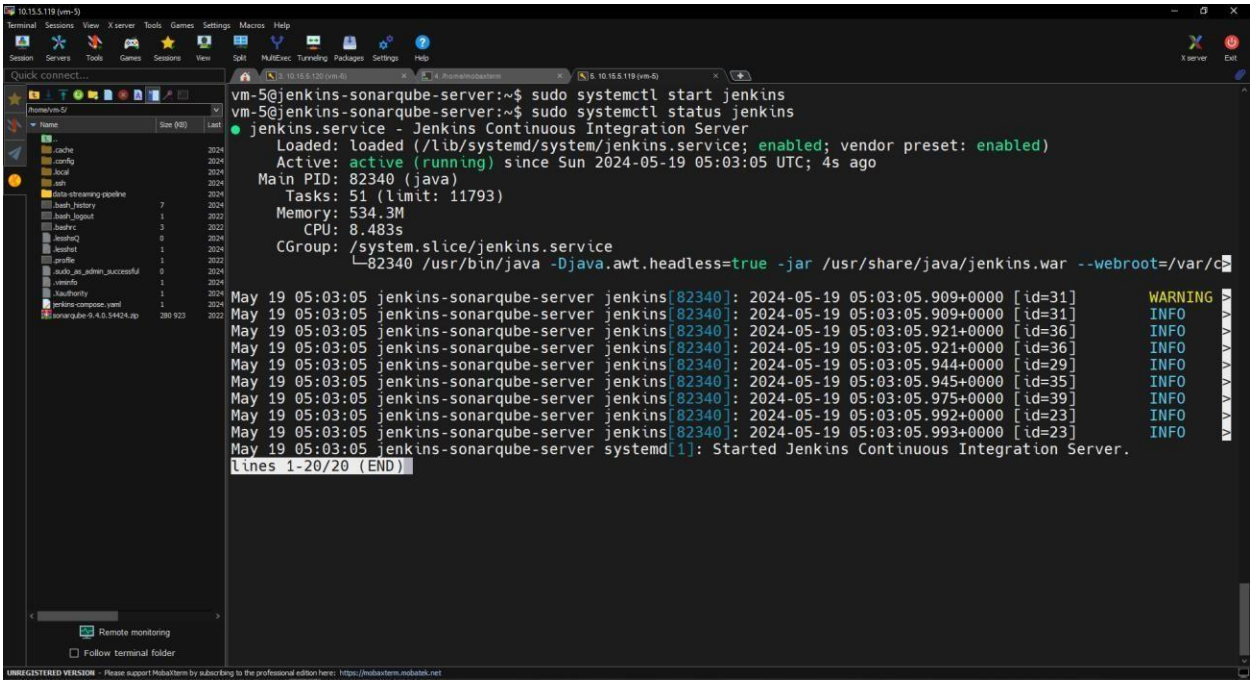


5.5.8 Check Prometheus Alerts Interface



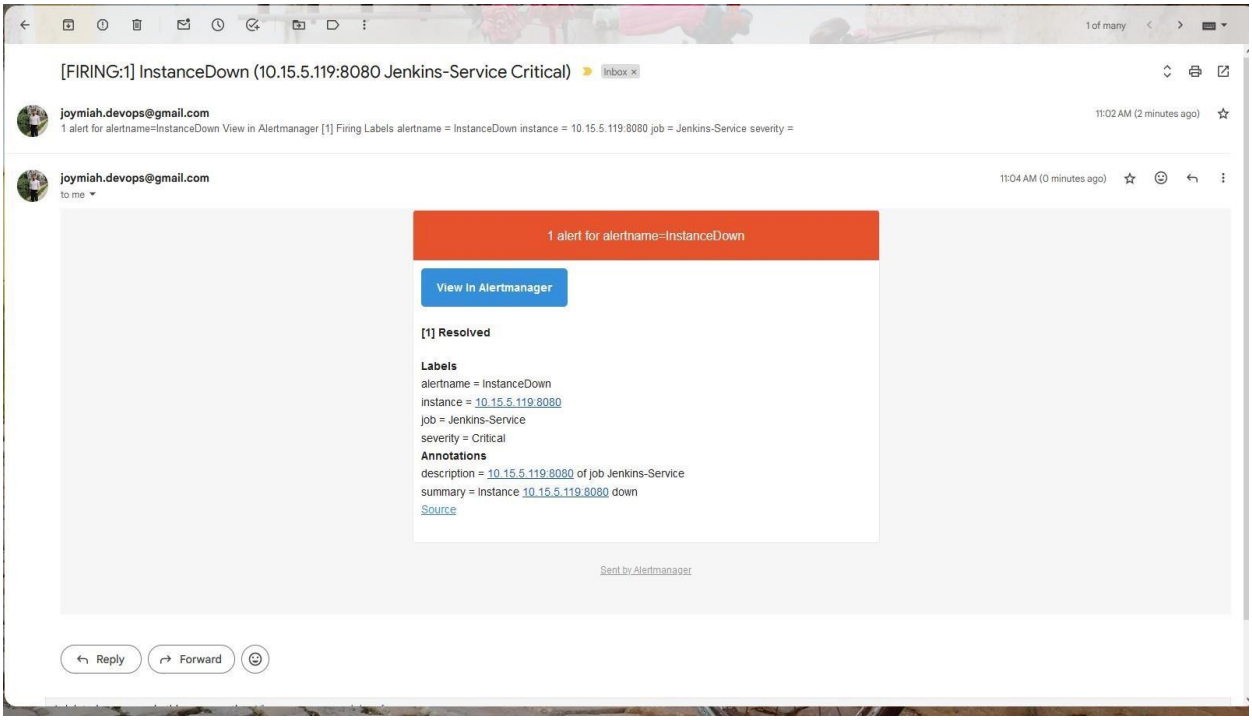
5.5.9 Check Email

6. Restart Jenkins: Restart the Jenkins service.



5.5.10 Start Jenkins

7. Alert Resolution: In the Alerts section, the instance will be marked as inactive again. After approximately two minutes, you will receive a resolved email indicating that the issue has been addressed.

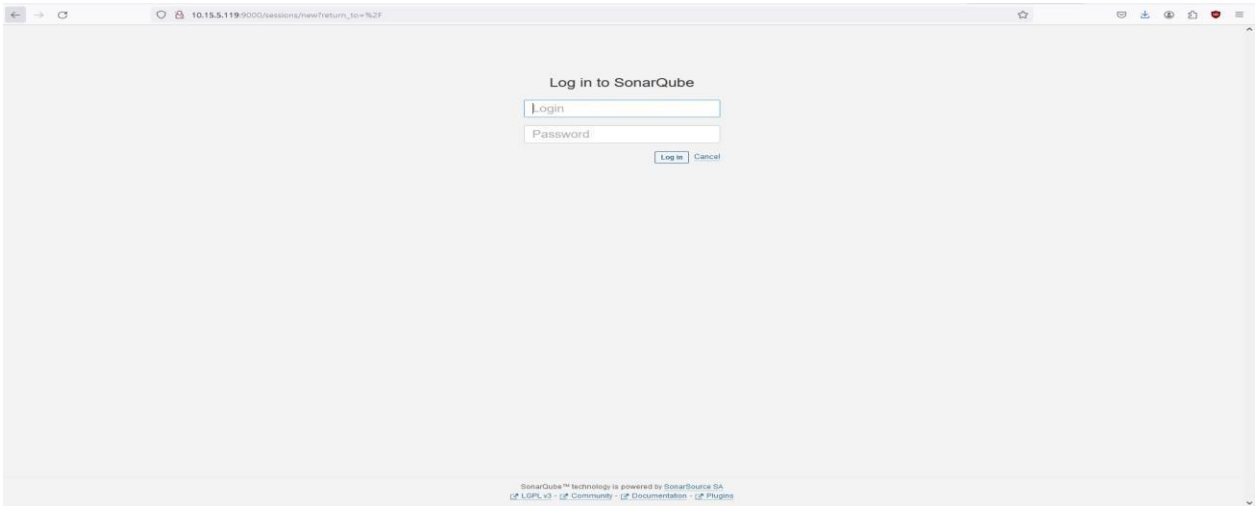


5.5.11 Check Resolved Email

5.6 Implementation of Sonarqube Analysis And Trivy

The implementation of SonarQube Analysis and Trivy:

SonarQube Analysis: There is an additional step in the ci/cd pipeline SonarQube, although it is an open-source. Continuously inspecting code quality. These are tasks done after the application had been created. Then I performed a static code analysis to check for possible variable bugs, code smells and security issues. At our System level, we have been running our Sonarqube as docker container.



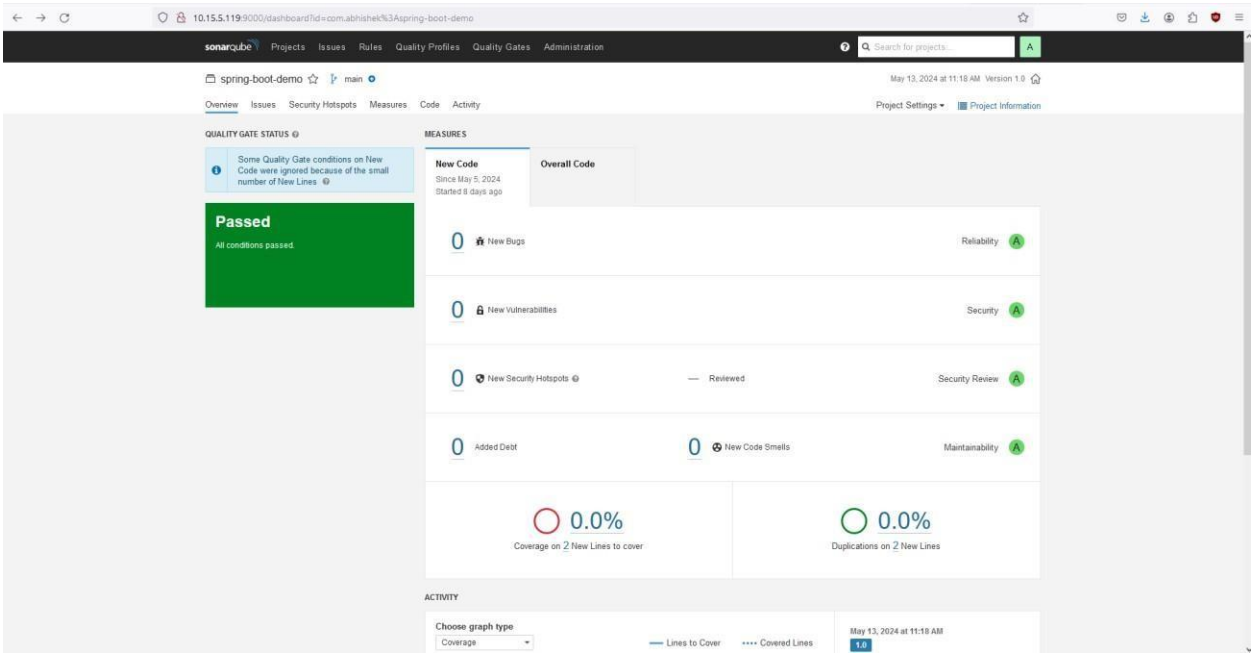
5.6.1 Sonarqube Login Page

Trivy Image Scanning: In the case of the just developed application, scan the docker image was done with Trivy which is a simple container vulnerability scanner. After the completion of audit using SonarQube, the docker the image was also run through Trivy to check for its vulnerabilities. Such tools were helpful in ensuring that the application was secure throughout the stages of development. By detecting and correcting any potential concerns that might arise early in the process of development. Which assisted in avoiding security defects from reaching the production line Stage. This enhanced overall efficiency and effectiveness of the system in functioning & trustworthiness.

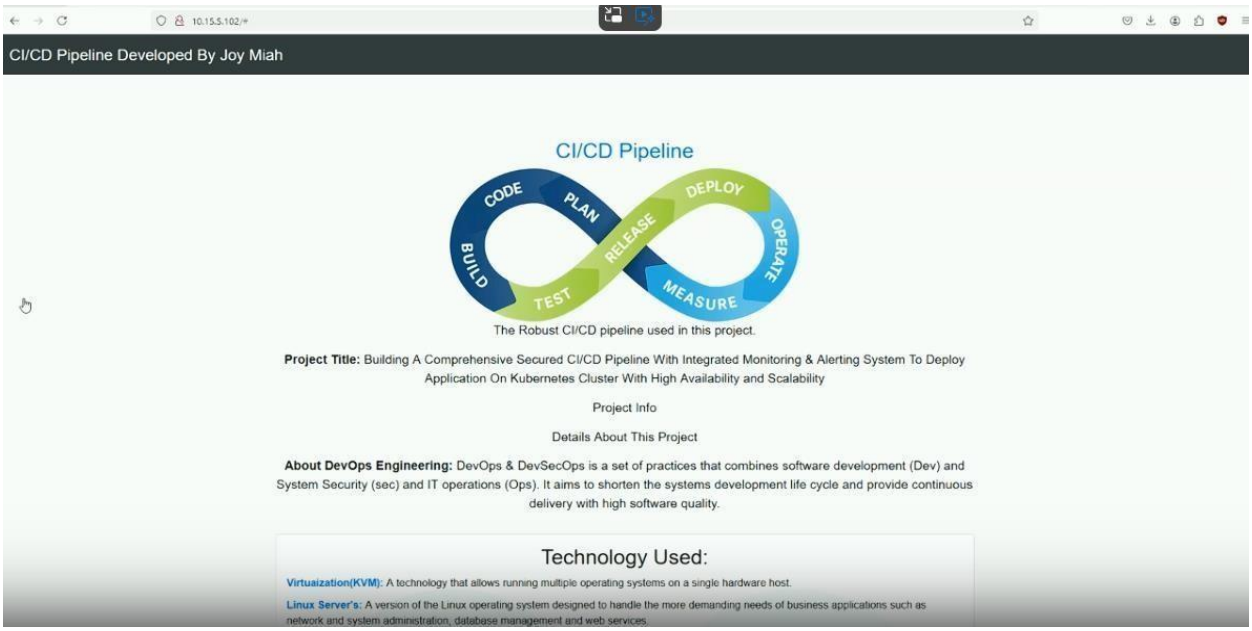
5.7 Testing of CI/CD Pipeline

The CI/CD pipeline was tested according to the following procedure:

After the build stage, unit tests for each part of the application were carried out to check for correct functionality and ensure components were functioning properly. Using unit testing, any bugs that would have been encountered were handled before moving to the next step in the pipeline. The next step was to conduct integration tests that would test various components of the system and their functioning. If there are any more mistakes observed during integration testing, these are rectified. Load testing was conducted to confirm that the application delivered the needed performance standards. This meant having to test the application at normal use and high utilization to take the load pressure off. This is a form of testing that is used to identify any openings that the attacker may use to compromise an application. This equaled the process of checking docker image for vulnerabilities by Trivy and analyzing static code with SonarQube tool. Deployment tests were also correctly verified by detect metrics. All these tests were very important. By as they distinguishing possible difficulties at the preliminary stage. They were able to contribute to preventing issues from getting to the production stage where they might cause even bigger problems.



5.7.1 SonarQube Analysis Report



5.7.2 Spring-Boot Application Interface

5.8 Testing of Scalability and High Availability

Scalability and high availability were tested in this way:

The CI/CD pipeline and Kubernetes cluster's scalability were tested to understand the network, application and manage escalating workloads. This also required the increase of the system load. In terms volume of traffic that is transmitted/received and the system could handle capacity or accommodating incremental demand without the quality easing.

High Availability Testing: As for HA (high availability) testing were done in order to ensure that application remained operational after a failure even if affected by it. This simulating challenges like node failure, network failure among others and that the applications should work as intended despite the failures.

Automatic Scaling Testing: To scale the cluster the initial right size was determined and then automatic scaling tests were performed could increase or decrease the total number of processors to meet the requirements based on loading conditions. This involved enhancement and reduction of the system's load and it also means that the number that was involved in pods increased and decrease proportionately. These tests were also important being used to determine how the system would perform against lot of work and ensure through load balancing.

CHAPTER 6

IMPACT & SUSTAINABILITY

6.1 Impact on Society

The project is titled **Building a Comprehensive Secured CI/CD Pipeline with Integrated Monitoring**. Here involves the practical execution of various systems, technologies, and tools that constitute the CI/CD pipeline & alerting system. Building extending Kubernetes cluster to run the application with high availability and scalability. It is a topic that has significant social implications, especially about the creation of software deployment.

Efficiency of Software Development: Automating the process of constructing, testing and deploying applications which reduces their complexity while enhancing their scalability, maintenance, and security. In such applications, the implementation of the project reduces the time and energy. This efficiency can help in the fast delivery of applications.

Enhanced Security: The incorporation of SonarQube as well as Trivy into the pipeline provides shielding to applications from potential risks. This not only guarantees application integrity but user's data, which ensures a safer digital environment.

Reliability and Availability: Combined with high availability through Kubernetes and other familiar UI controls, the result is very smooth and guarantees that the availability and scalability options make applications always available for use by the end user. It enhances user satisfaction and ensures confidence while using the apps in the digital setting.

Monitoring and Alerting: Prometheus and Grafana for live observation of the effectiveness of the models, as well as establishing an alerting mechanism that can be used to detect existing problems and address them before they become a big issue. This leads to of better solutions as well as much more stable and trustworthy applications that enhances user's satisfaction.

Finally, this project advances software development processes by increasing efficiency, security and reliability. This has a positive impact on society, as software applications become more integrated into our daily lives.

Educational Value: The lack of well-structured guidance available for architects to follow when constructing a CI/CD pipeline. This information could be very valuable for students, teachers and professionals who develop software.

Last of all, this project enhances software development processes through elevating the speed and safety of proactive algorithms. This has a beneficial effect for society.

6.2 Ethical Aspects

Alerting System: High availability services to ensure the deployment of the application on a Kubernetes cluster follows numerous ethical principles:

Data Privacy and Security: The security of the applications are the main areas of concern in the project process. Therefore, adding SonarQube and trivy in the CI/CD pipeline gives a guarantee that the developed applications should be considered the protection of the data. This dedication add the additional benefit of privacy and security compliance with the ethical ideal respect for privacy.

Reliability and Trustworthiness: Therefore, the project's objective is to design a CI/CD pipeline that integrates all the dependencies reliable and trustworthy. The project makes sure that the installed equipment is highly available and can grow in terms of its size. It has offered loyal services to the users in its various applications. This commitment to dependability and specialization is dependent on this technique as the keystone of their identification system trustworthiness. Which is one of the basic ethical principles. Transparency is possible due to the occurrence of the project whereby incidents detected are immediately monitored and alerted and constructed continuous integration/continuous delivery (CI/CD) pipeline and Kubernetes cluster. This assists stakeholders to develop a novel and much-appreciated perspective of the system.

6.3 Sustainability Plan

The specific goals to achieve the identified objectives by implementing the project's sustainability plan namely **Building a Comprehensive Secured CI/CD Pipeline with Integrated Monitoring & Alerting System** is as follows:

Continuous Improvement: This project is designed as a work in progress and constructed with the ability to update when necessary. This involves the update of the CI/CD pipeline stages which include new tools. It characteristic includes taking the necessary actions in order to advance the corresponding field technologies and optimize the monitoring and alarm system.

Resource optimization: The project aims to reduce waste and the consumption of resources and energy. Implementation of Kubernetes to organization ensures the efficient use of available resources consume, which is beneficial for the subsequent continuation of the project. Which concerns the issue of project sustainability.

Scalability: The idea is to develop the project such that it can sustain the further load. More importantly, within this scalability and capabilities helps to ensure the company is able to provide enough delivery of services.

Maintainability: The project was to be sufficiently manageable so that maintenance would be straightforward. It involve going deeper and deeper into details & documentation with well-structured, well-coded and user friendly interface. This makes it easy to update and adapt to new workings operations through the process. It also helps in enhancing sustainability of the project because various measures are put in place to ensure that their budget is well managed.

Security upgrades: Frequency of security upgrade is not clear but its aim is to safeguard the applications from any threats. This also enable creating new SonarQube and Trivy scans in the database as well as changing their details frequently docker image and code.

Training and Knowledge Transfer: Training development and dissemination are essentially sessions to ensure that organizational team members continue to enjoy the latest technology and enhanced techniques. This makes sure that the project may well progress and take the needed changes. Thus, increasing possible sustainability. In conclusion, with the help of this sustainability strategy, one may ensure that the project is very useful.

CHAPTER 7

CONCLUSION & FUTURE SCOPE

7.1 Discussion & Conclusions

With the rapid advancement in software development, application delivery has become an essential area of concern for organizations.

Thus, this project effectively created and enhanced an automated CI/CD pipeline that eliminates as much of the human factor as possible. Updating the status as frequently as possible makes the pipeline extremely helpful in deployment. Some of the milestones are the creation of an efficient CI/CD loop to shorten the time to deploy the applications and including tools to monitor the application's performance in real-time. Proactive alerting specifically, which is grounded on predetermined rule sets, assists that issues become resolved quickly. To complement testing a self-contained optimistic security check keeps the overall code caliber high and Identifies security threats.

The project also fully considers high availability and scalability which ensure uninterrupted service. Introducing the automated rollbacks improves deployment reliability and provides a fast fix if there are issues.

Thus, the aim of the project to increase the speed and reliability of the software and ensure the security. This is very effective. The successful integration of this CI/CD pipeline and monitoring system leaves a good starting point for future projects as well as continual enhancement of the software delivery processes.

7.2 Scope for Further Developments

The project is called a Building the right and secured CI/CD pipeline with integrated monitoring & Alerting Systems. Here how it could be investigated:

Enhanced Security Measures: There is even more improvement as the top-tier security scanning tools and truly implemented security procedures like static and dynamic application security testing. Implement more comprehensive security policies and ensure compliance with industry standards in such a way that they are updated and cover all the regulated aspects.

Machine Learning and AI Integration: It is also possible to engage and apply machine learning and AI in order to identify situations which can become problematic. There is a potential to use big data and analytics to gain better understanding of the application usage, issues, and security threats.

Advanced Monitoring and Alerting: Introduce higher level instruments and methods in order to increase the monitoring capacity. Use predictive analysis and real time anomalies to recognize performance and security problems before they occur.

Multi-Cloud Deployment: Introduce the way of deployment that allows using the tool both in the multi-cloud and the hybrid-cloud modes. This is done to increase the availability and redundancy at different geographical location and cloud service provider.

Scalable Micro services Architecture: Further break down application into smaller services for improving scalability, modularity and ease of deployment. Launch the service mesh technologies like Istio so as to control the communication between the micro services.

REFERENCES

[1] A. Balalaie, A. Heydarnoori, and P. Jamshidi, "Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture," IEEE Software, vol. 33, no. 3, pp. 42-52, May-June 2016.

[2] M. Shahin, M. Ali Babar, and L. Zhu, "Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices," IEEE Access, vol. 5, pp. 3909-3943, 2017.

[3] L. Bass, I. Weber, and L. Zhu, "DevOps: A Software Architect's Perspective," IEEE Software, vol. 32, no.2, pp. 12-15, Mar.-Apr. 2015.

[4] J. Humble and J. Molesky, "Why Enterprises Must Adopt Devops to Enable Continuous Delivery," Cutter IT Journal, vol. 24, no. 8, p. 6, 2011

[5] CNCF, "Cloud Native Computing Foundation." [Online]. Available: https://www.cncf.io/. [Accessed: 30-Jun- 2024].

[6] JOYMIAH, "CI/CD Pipeline DIU," GitHub. [Online]. Available: https://github.com/JOYMIAH/CI-CD-Pipeline-DIU.git. [Accessed: 30-Jun-2024].

[7] Kubernetes. [Online]. Available: https://kubernetes.io/. [Accessed: 30-Jun-2024].

[8] Docker Hub. [Online]. Available: https://hub.docker.com/. [Accessed: 30-Jun-2024].

[9] Docker. [Online]. Available: https://www.docker.com/. [Accessed: 30-Jun-2024].

[10] AWS. [Online]. Available: https://aws.amazon.com/console/. [Accessed: 30-Jun-2024].

[11] Google Cloud. [Online]. Available: https://console.cloud.google.com/welcome?project=rock-idiom-345210. [Accessed: 30-Jun-2024].

[12] ArgoCD. [Online]. Available: https://argo-cd.readthedocs.io/en/stable/. [Accessed: 30-Jun-2024].

[13] Prometheus. [Online]. Available: https://prometheus.io/. [Accessed: 30-Jun-2024].

[14] Grafana. [Online]. Available: https://grafana.com/. [Accessed: 30-Jun-2024].

[15] Jenkins. [Online]. Available: https://www.jenkins.io/. [Accessed: 30-Jun-2024].

[16] Alert-Manager. [Online]. Available: https://prometheus.io/docs/alerting/latest/alertmanager/. [Accessed: 30-Jun-2024].

[17] SonarSource, "SonarQube." [Online]. Available: https://www.sonarsource.com/products/sonarqube/. [Accessed: 30-Jun-2024].

[18] Red Hat, "What is KVM?" [Online]. Available: https://www.redhat.com/en/topics/virtualization/what-is-KVM. [Accessed: 30-Jun-2024].

[19] Trivy. [Online]. Available: https://trivy.dev/. [Accessed: 30-Jun-2024].

212-15-4108.pdf

ORIGINALITY REPORT

10%	8%	3%	7%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	dspace.daffodilvarsity.edu.bd:8080 Internet Source	4%
2	Submitted to Oklahoma State University Student Paper	1%
3	Submitted to Daffodil International University Student Paper	1%
4	medium.com Internet Source	<1%
5	estudogeral.sib.uc.pt Internet Source	<1%
6	Submitted to Grambling State University Student Paper	<1%
7	Submitted to University of Waikato Student Paper	<1%
8	compass.2i2c.org Internet Source	<1%
9	Submitted to Universitaet Paderborn Student Paper	<1%