Project Report On

Integrating AI and Automation for a secure CI/CD pipeline deployment

Submitted

In partial fulfilment

For the award of the Degree of

Post Graduate Diploma in IT Infrastructure System & Security

CDAC(Pune)

Guided by:
Mr. Suyash Kulkarni

Submitted by:
Aditya Kumar PRN: 250840123003
Afreen Ansari PRN: 250240123004
Ashish Ranjan PRN:250240123007
Jayesh Vispute PRN: 250240123021
Htay Htay Yi PRN: 250240123061

Centre of Development of Advanced Computing (C-DAC)

ACTS (Pune – 411008)

CERTIFICATE

TO WHOMSOEVER IT MAY CONCERN

This is to certify that

Aditya Kumar	PRN: 250840123003
Afreen Ansari	PRN: 250240123004
Ashish Ranjan	PRN: 250240123007
Jayesh Vispute	PRN: 250240123021
Htay Htay Yi	PRN: 250240123061

Have successfully completed their project on

“ Integrating AI and Automation for a secure CI/CD pipeline deployment ”

Under the guidance of Mr. Suyash Kulkarni

Project Guide

Mr. Suyash Kulkarni

Course Coordinator

Ms. Divya Patel

HOD ACTS

Mr. Gaur Sunder

ACKNOWLEDGEMENT

This is to acknowledge our indebtedness to our Project Guide, **Mr. Suyash Kulkarni**, CDAC ACTS, Pune for his constant guidance and helpful suggestions for preparing this project **Integrating AI and Automation for a production grade secure CI/CD pipeline deployment.**

We express our deep gratitude towards his inspiration, personal involvement, constructive criticism that he provided us along with technical guidance during the course of this project.

We take the opportunity to thank head of department **Mr. Gaur Sunder** for providing us such a great infrastructure and environment for our overall development.

We express sincere thanks to **Mr. Soumyakant Behera**, Process Owner, for their kind cooperation and their extendible support towards the completion of our project.

It is our great pleasure in expressing sincere and deep gratitude towards **Mrs. Risha P.R.** (Program Head) and **Ms. Divya Patel**(Course Coordinator, PG-DITISS) for their valuable guidance and constant support throughout this work and help to pursue additional studies.

Also, our warm thanks to **C-DAC ACTS** Pune, which provided us this opportunity to carry out this prestigious Project and enhance our learning in various technical fields.

Aditya Kumar	PRN: 250840123003
Afreen Ansari	PRN: 250240123004
Ashish Ranjan	PRN: 250240123007
Jayesh Vispute	PRN: 250240123021
Htay Htay Yi	PRN: 250240123061

ABSTRACT

This project presents the design and implementation of an automated and secure CI/CD pipeline while proposing the integration of Artificial Intelligence (AI) to enhance security and efficiency in software deployment. The pipeline automates critical stages such as code integration, build, testing, code quality analysis, containerization, and deployment using modern DevOps tools. Security is incorporated through automated vulnerability scanning and quality checks to ensure reliable releases. Although AI is not implemented in the current phase, the project outlines a future-ready framework where AI can be used for intelligent threat detection, anomaly analysis, and automated decision-making, aligning the solution with emerging DevSecOps practices in corporate environments.

TABLE OF CONTENTS

1.INTRODUCTION AND OVERVIEW OF PROJECT

1.1 PURPOSE

The purpose of this project is to design and implement an **automated CI/CD pipeline** for efficient software delivery. It aims to integrate **security practices** throughout the pipeline to ensure secure application deployment.The project focuses on **reducing manual intervention** and deployment errors using automation tools. It also demonstrates the use of **containerization and orchestration** for scalable deployments. Additionally, the project integrates AI-assisted security analysis using the Gemini API to enhance vulnerability interpretation, prioritization, and intelligent decision-making.

1.2 AIM AND OBJECTIVE

1. **Aim**: The aim of this project is to develop a **secure, automated, and intelligent CI/CD pipeline** for continuous software delivery by integrating **DevSecOps practices and workflow automation using n8n**. The project focuses on embedding security, quality analysis, monitoring, event-driven automation, and AI-assisted vulnerability analysis throughout the deployment lifecycle to achieve a scalable and production-ready DevSecOps solution.

2. **Objective**: The project seeks to achieve the following objectives:

- **Automated CI/CD Pipeline:**
  Design and implement a fully automated CI/CD pipeline using **Jenkins** to enable continuous integration, testing, security validation, and deployment of the application.

- **Containerization:**
  Utilize **Docker** to containerize the application, ensuring consistency, portability, and reproducibility across development, testing, and production environments.

- **Kubernetes-Based Deployment:**
  Deploy the application on a **Kubernetes cluster** to enable efficient container orchestration, scalability, fault tolerance, and high availability.

- **Source Code Management:**
  Implement **Git and GitHub** for version control, collaboration, and automatic pipeline triggering through webhooks.

- **Code Quality and Security Analysis:**
  Integrate **SonarQube** for static code analysis to ensure high code quality and early detection of bugs, vulnerabilities, and code smells.

- **Vulnerability Scanning:**
  Use **Trivy** to perform file system and container image vulnerability scanning, preventing insecure artifacts from reaching the deployment stage.
- **AI-Assisted Vulnerability Analysis:**

  Utilize the Gemini API to analyze, summarize, and prioritize CVE reports generated by Trivy, enabling faster and more intelligent security decision-making.

- **Artifact Management:**
  Utilize **Nexus Repository Manager** to securely store, manage, and version build artifacts and container images.

- **Monitoring and Observability:**
  Integrate **Prometheus and Grafana** for real-time monitoring, visualization, and performance analysis of the application and infrastructure.

- **Workflow Automation with n8n:**
  Implement **n8n** to automate event-driven workflows such as build notifications, security alerts, pipeline status handling, and integration between CI/CD, monitoring, and communication systems.

- **Intelligent and AI-Assisted Automation:**
  Leverage AI-assisted analysis and automated workflows using n8n and the Gemini API to support intelligent security insights, proactive alerting, and future-ready DevSecOps operations.

2.PREREQUISITES

2.1 Hardware and Software Requirements

Hardware Requirements

1. Machine 1 – DevOps & CI/CD Server (Jenkins, SonarQube, Nexus, Docker, Trivy, n8n)

- **RAM:** Minimum 4 GB
- **CPU:** 2 Cores
- **Disk Space:** Minimum 20 GB

2. Machine 2 – Kubernetes Cluster (Master + Worker Nodes):

- **RAM:** Minimum 8 GB
- **CPU:** 2 Cores per node
- **Disk Space:** Minimum 40 GB

3. Machine 3 – Monitoring Server (Prometheus, Grafana, Blackbox Exporter):

- **RAM:** Minimum 4 GB
- **CPU:** 2 Cores
- **Disk Space:** Minimum 20 GB

Software Requirements

1. **Operating System:** Ubuntu 20.04 LTS or any Linux-based distribution
2. **Version Control:** Git and GitHub
3. **Programming Language:** Java
4. **Framework:** Spring Boot
5. **Build Tool:** Apache Maven
6. **CI/CD Tool:** Jenkins
7. **Container Platform:** Docker
8. **Container Orchestration:** Kubernetes (kubeadm, kubelet, kubectl)
9. **Code Quality Tool:** SonarQube
10. **Security Scanning Tool:** Trivy
11. **Artifact Repository:** Nexus Repository
12. **Monitoring Tools:** Prometheus, Grafana, Blackbox Exporter
13. **Workflow Automation Tool:** n8nAI Integration: Gemini API (for AI-assisted CVE analysis and vulnerability prioritization)

2.1 TOOLS AND TECHNOLOGIES

A) CI/CD, Automation, and AI-Driven Deployment Tools:

1. Jenkins

- Installed and configured on the **CI/CD server**.
- Used to automate the complete CI/CD pipeline including build, testing, security scanning, containerization, and deployment.
- Required plugins include **Git**, **Docker Pipeline**, **SonarQube Scanner**, **Email Extension**, and **Trivy integration**.
- A **Jenkinsfile** is used to define the declarative pipeline stages.

2. n8n (Workflow Automation Tool)

- Gemini API (AI-Assisted Security Analysis)
- Used to analyze, summarize, and prioritize CVE reports generated by Trivy.
- Provides AI-driven insights into vulnerability severity, impact, and remediation guidance.
- Integrated through n8n workflows to automate intelligent security reporting and alerts.

3. Docker & Docker Registry

- Docker is used to containerize the Spring Boot application.
- Docker images are pushed to **Docker Hub** and pulled during deployment.
- Ensures consistency across environments.

4. Kubernetes

- A Kubernetes cluster (1 Master + Worker Nodes) is used for application deployment.
- Kubernetes manages container orchestration, scalability, and service exposure.
- `kubectl` is configured on the Jenkins server to trigger deployments.

5. Nexus Repository

- Used as an artifact repository to store and manage Maven build artifacts securely.

6. Monitoring Tools (Prometheus & Grafana)

- Prometheus is used to collect system and application metrics.
- Grafana is used to visualize metrics and monitor system health.
- Blackbox Exporter is used for service availability monitoring.

B) Security and Code Quality Tools:

1. SonarQube

- Integrated into the Jenkins pipeline for **static code analysis (SAST)**.
- Ensures code quality, detects bugs, vulnerabilities, and code smells.
- Quality Gates are used to evaluate build status.

2. Trivy

- Used for **file system scanning** and **Docker image vulnerability scanning**.
- Helps identify security risks before deployment

3. Gemini API

- Used for AI-assisted interpretation and prioritization of vulnerability (CVE) reports.
- Enhances security decision-making and reduces manual analysis effort.

C) Networking and Access Control

- Ensure proper communication between the **Jenkins server** and the **Kubernetes cluster** by configuring firewall rules and security groups.
- GitHub must be accessible for Jenkins to pull source code and trigger pipelines.
- Docker Hub access is required for pushing and pulling container images securely.

D) Basic Configuration Knowledge

- **Git & GitHub:** For source code management and version control.
- **Jenkins Pipeline:** Understanding Jenkinsfile syntax for automating CI/CD workflows.
- **Docker:** Ability to write Dockerfiles for application containerization.
- **Kubernetes:** Knowledge of Kubernetes resources such as pods, deployments, services, and namespaces.
- **Monitoring Tools:** Basic understanding of Prometheus metrics collection and Grafana dashboards.
- **n8n:** Designing workflow-based automation and event-driven integrations.

E) Access Credentials

- **GitHub Credentials:** GitHub Personal Access Token or SSH key configured in Jenkins.
- **Docker Hub Credentials:** Docker registry credentials stored securely in Jenkins.
- **Kubernetes Credentials:** `kubeconfig` file configured to allow Jenkins to deploy applications to the cluster.
- **n8n Credentials:** Webhooks, API tokens, and service credentials securely configured for workflow execution.

F) Security Considerations

- **SonarQube Quality Gates** are enforced to maintain code quality and security standards.
- **Trivy Scanning** ensures vulnerabilities are detected early in the pipeline.
- **Monitoring Alerts** are configured using Prometheus to track system performance and failures.
- **Email Notifications** are enabled in Jenkins to notify build and deployment status.

3. OVERALL DESCRIPTION

3.1 INTRODUCTION:

In today's rapidly evolving software development environment, organizations require fast, reliable, and secure application delivery mechanisms to remain competitive. Traditional manual deployment approaches are time-consuming, error-prone, and often lack adequate security controls. To address these challenges, modern software teams adopt **Continuous Integration and Continuous Deployment (CI/CD)** practices combined with **workflow automation** to streamline the build, test, security, and deployment processes.

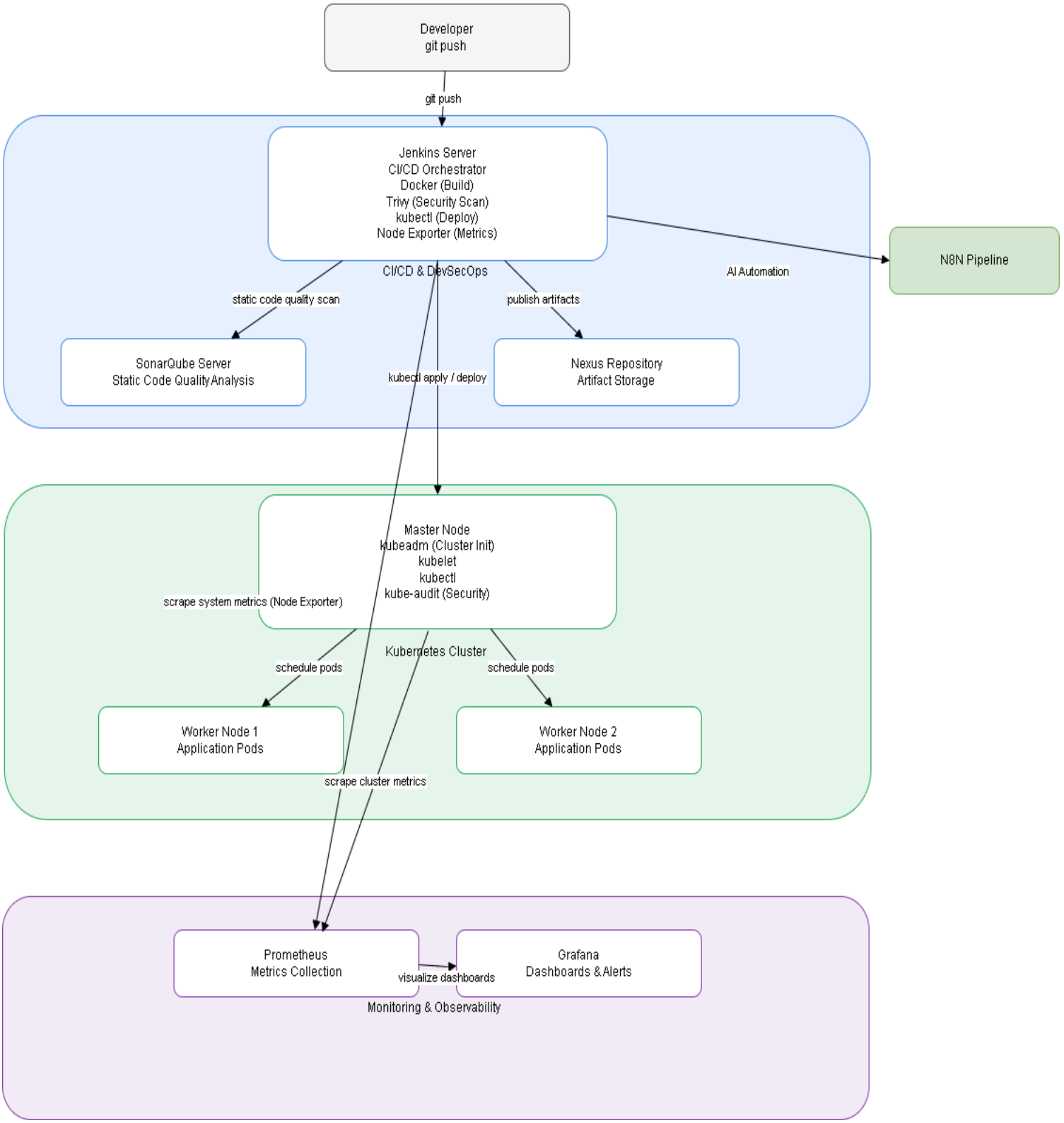
This project, titled **"Integrating AI and Automation for a Secure CI/CD Pipeline Deployment,"** focuses on designing and implementing a **secure, automated, and event-driven CI/CD pipeline** for deploying a Spring Boot–based application in a Kubernetes environment. The solution utilizes **Jenkins** to orchestrate the CI/CD workflow, **Docker** for containerization, Kubernetes for container orchestration, n8n for workflow automation, and the Gemini API for AI-assisted security analysis, enabling scalable, consistent, and intelligent pipeline execution.

To enhance software quality and security, the pipeline integrates **SonarQube** for static code analysis and **Trivy** for vulnerability scanning of file systems and container images. Additionally, **Nexus Repository Manager** is used to securely manage and version build artifacts. For operational visibility and observability, **Prometheus** and **Grafana** are implemented to continuously monitor system performance, resource utilization, and application health in real time.

The CI/CD pipeline is triggered automatically through **GitHub webhooks**, ensuring continuous integration upon code changes. n8n workflows are used to automate event-driven actions such as pipeline notifications, security alerts, AI-assisted CVE analysis using the Gemini API, and integration between CI/CD and monitoring components.

The project integrates AI-assisted security analysis using the Gemini API to analyze and prioritize vulnerability (CVE) reports generated during the CI/CD process. This enhances security decision-making, reduces manual analysis effort, and strengthens the overall DevSecOps posture of the pipeline.

3.2 DATA FLOW DIAGRAM



The workflow begins when a developer pushes code to the GitHub repository, which automatically triggers the Jenkins CI/CD pipeline. Jenkins performs build, testing, static code analysis using SonarQube, and vulnerability scanning with Trivy, followed by Docker image creation and artifact storage in the Nexus Repository. The validated application is then deployed to the Kubernetes cluster using kubectl, where the master node schedules pods across worker nodes for scalable and fault-tolerant execution. Prometheus continuously collects system and application metrics, which are visualized through Grafana dashboards for monitoring and alerting. Pipeline events, security scan results, and monitoring alerts are forwarded to the n8n automation pipeline, enabling AI-assisted analysis, notifications, and event-driven workflow automation, making the overall CI/CD process secure, intelligent, and production-ready.

3.3 ARCHITECTURE

The architecture of the system is designed to enable a **secure, automated, and scalable CI/CD pipeline** through the integration of DevOps and DevSecOps tools. The architecture ensures continuous integration, automated deployment, AI-assisted security validation, workflow automation, and real-time monitoring of the application.

Continuous Integration with Jenkins

- **Triggering Builds:** Code changes pushed to the GitHub repository automatically trigger the Jenkins pipeline.
- **Build and Test:** Jenkins performs source code checkout, compilation, unit testing, and static code analysis using **SonarQube** to maintain code quality and security standards.
- **Failure Reporting:** In case of build or test failures, Jenkins generates detailed logs and reports, enabling developers to quickly identify and resolve issues.

Docker Image Creation

- **Successful Builds:** After successful validation, Jenkins packages the application and creates a **Docker image**.
- **Artifact Storage:** Build artifacts are published to the **Nexus Repository** for secure storage and version management.
- **Security Scanning:** The Docker image and file system are scanned using **Trivy** to detect vulnerabilities before deployment.

AI-Assisted Security Analysis and Workflow Automation

- After vulnerability scanning using Trivy, the generated CVE data is forwarded to n8n through a webhook-based integration. n8n processes this data and invokes the Gemini API to perform AI-assisted analysis, summarization, and prioritization of vulnerabilities.
- The AI-generated security insights are then used to automate alerts and reporting, enabling faster and more intelligent security decision-making.

Deployment to Kubernetes

- **Kubernetes Integration:** Jenkins deploys the Docker image to the Kubernetes cluster using `kubectl`.
- **Orchestration and Scaling:** The Kubernetes master node schedules application pods across multiple worker nodes, ensuring scalability, high availability, and fault tolerance.

Monitoring and Reporting

- **Performance Monitoring: Prometheus** collects system and application metrics from the Jenkins server and Kubernetes nodes.
- **Visualization and Alerts: Grafana** displays metrics through dashboards and enables alerting for proactive monitoring.
- **Pipeline Notifications and Automation:** Pipeline Notifications and Automation: Jenkins forwards pipeline events and security scan results to n8n, which automates notifications and alerts based on AI-assisted vulnerability analysis performed using the Gemini API.

4. CI/CD LAB SETUP

To set up a **production-grade DevSecOps CI/CD pipeline**, this lab uses a **distributed 7-machine architecture**, where each component is deployed on a dedicated server. This approach closely mirrors real-world enterprise DevSecOps environments and avoids single-machine limitations.

Machine 1 – Kubernetes Master Node

- Kubernetes control plane
- kubeadm, kubelet, kubectl
- kubeaudit for Kubernetes security auditing

Machine 2 – Kubernetes Worker Node 1

- Runs application pods
- kubeadm, kubelet

Machine 3 – Kubernetes Worker Node 2

- Runs application pods
- kubeadm, kubelet

Machine 4 – Jenkins CI Server

- Jenkins for CI/CD orchestration
- Docker for container image creation
- Trivy for vulnerability scanning
- kubectl for Kubernetes deployment
- n8n for workflow automation and AI integration
- Gemini API for AI-assisted CVE analysis

Machine 5 – SonarQube Server

- SonarQube for static code analysis (SAST)
- Docker-based SonarQube deployment

Machine 6 – Nexus Repository Server

- Nexus Repository Manager
- Docker-based artifact storage

Machine 7 – Monitoring Server

- Prometheus
- Grafana
- Blackbox Exporter

4.2 Jenkins CI Server Setup (Machine 4)

4.2.1 Install Docker

Docker is installed on the Jenkins server to build and manage container images.

```
sudo apt update
sudo apt install -y docker.io
sudo systemctl start docker
sudo systemctl enable docker
sudo usermod -aG docker $USER
```

Verify installation:

```
docker --version
```

4.2.2 Install Jenkins

Jenkins is installed to orchestrate the CI/CD pipeline.

```
sudo apt update
sudo apt install -y openjdk-17-jdk
wget -q -O - https://pkg.jenkins.io/keys/jenkins.io.key | sudo apt-key add -
sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable /etc/apt/sources.list.d/jenkins.list'
sudo apt update
sudo apt install -y jenkins
sudo systemctl start jenkins
sudo systemctl enable jenkins
```

Access Jenkins at:
http://<Jenkins-Server-IP>:8080

4.2.3 Install Jenkins Plugins

The following plugins are installed via **Manage Jenkins → Manage Plugins**:

- Git Plugin (GitHub integration)

- Docker Pipeline Plugin
- SonarQube Scanner for Jenkins
- Kubernetes CLI Plugin
- Email Extension Plugin

4.2.4 Install Trivy

Trivy is installed on the Jenkins server for vulnerability scanning.

```
wget
https://github.com/aquasecurity/trivy/releases/latest/download/trivy_0.xx.x_Linux-64bit.deb
sudo dpkg -i trivy_0.xx.x_Linux-64bit.deb
```

4.2.5 n8n Workflow Automation and AI Integration

- n8n is deployed on the Jenkins CI server to enable event-driven automation and AI-assisted security analysis. Jenkins forwards vulnerability scan results to n8n via webhooks.
- n8n processes CVE data and integrates with the Gemini API to analyze, summarize, and prioritize vulnerabilities. Based on AI-generated insights, n8n automates notifications and security alerts.

4.3 SonarQube Setup (Machine 5)

SonarQube is deployed as a Docker container to perform **static code analysis**.

```
docker run -d --name sonarqube -p 9000:9000 sonarqube:community
```

Access SonarQube at:
http://<SonarQube-IP>:9000

Jenkins is configured to send code analysis reports to SonarQube, and **quality gates** are enforced before pipeline progression.

4.4 Nexus Repository Setup (Machine 6)

Nexus Repository Manager is deployed using Docker to manage build artifacts.

```
docker run -d -p 8081:8081 --name nexus sonatype/nexus3
```

Responsibilities:

- Store build artifacts
- Host Docker images

- Manage dependency versions

4.5 Kubernetes Cluster Setup (Machines 1, 2, and 3)

4.5.1 Install Kubernetes Components

On all Kubernetes nodes:

```
sudo apt install -y kubeadm kubelet kubectl
sudo systemctl enable kubelet
```

- The **master node** initializes the cluster using kubeadm init
- Worker nodes join the cluster using the generated kubeadm join command
- kubectl is configured on the Jenkins server for deployment access

4.5.2 Kubernetes Security with kubeaudit

kubeaudit is used on the master node to validate Kubernetes manifests and detect:

- Privileged containers
- Missing resource limits
- Insecure RBAC permissions

4.6 Monitoring Stack Setup (Machine 7)

4.6.1 Prometheus

- Collects metrics from Kubernetes nodes and services
- Scrapes pod, node, and container metrics

4.6.2 Grafana

- Visualizes metrics collected by Prometheus
- Displays dashboards for CPU, memory, pod health, and node availability

4.6.3 Blackbox Exporter

- Performs endpoint and service health checks
- Monitors HTTP/TCP availability

4.7 CI/CD Pipeline Execution Flow

1. Developer pushes code to the GitHub repository
2. Jenkins pipeline is triggered automatically
3. Jenkins performs build and unit testing
4. SonarQube executes static code analysis
5. Trivy scans Docker images for vulnerabilities
6. Docker image is pushed to Nexus Repository
7. Kubernetes manifests are validated using kubeaudit
8. Application is deployed to the Kubernetes cluster
9. CVE scan results are forwarded to n8n for AI-assisted analysis using the Gemini API
10. n8n triggers automated notifications and security alerts based on AI analysis
11. Prometheus collects runtime metrics
12. Grafana visualizes system health and alerts

4.8 Final Outcome

This CI/CD lab setup delivers a **production-aligned DevSecOps pipeline** with:

- Distributed infrastructure
- Security enforced at every stage
- Kubernetes-native deployment
- Event-driven automation and AI-assisted security analysis
- Continuous monitoring and observability

5. WORKING

This section explains the end-to-end working of the production-grade DevSecOps CI/CD pipeline, detailing how code flows from development to deployment with integrated security, automation, and monitoring.

5.1 GitHub – Source Code Management Workflow

Workflow

- Developers push code changes to a **GitHub repository** that contains the Spring Boot application source code and pipeline configuration files.
- GitHub acts as the **single source of truth** for application code and CI/CD configuration.
- A webhook is configured in GitHub to notify Jenkins whenever a push event occurs.

Role in the CI/CD Pipeline

- Stores and manages application source code
- Maintains version control and collaboration
- Triggers the CI pipeline automatically via webhook integration

5.2 Jenkins – Continuous Integration Workflow

Workflow

- Jenkins listens for GitHub webhook events.
- Upon detecting a code change, Jenkins pulls the latest source code.
- The pipeline executes build and unit tests using Maven.
- Pipeline stages are defined using a **Jenkinsfile**, ensuring pipeline-as-code.
- Jenkins forwards pipeline execution status and vulnerability scan results to n8n through a webhook for automated processing.

Code Quality Analysis

- Jenkins integrates with **SonarQube** to perform static code analysis.
- SonarQube evaluates the code for:
  - Bugs
  - Code smells
  - Security vulnerabilities
  - Test coverage

Security Scanning

- Jenkins runs **Trivy file system scans** to analyze application dependencies.
- After containerization, **Trivy image scans** are performed to detect OS-level and application-level CVEs.

Role in the CI/CD Pipeline

- Automates build, test, and validation stages
- Enforces quality gates and security checks
- Prevents insecure or low-quality code from progressing
- Forwards CI/CD events and security scan data to n8n for AI-assisted automation and notifications.

5.3 Docker – Containerization Workflow

Workflow

- Jenkins uses Docker to build a container image of the Spring Boot application.
- The Dockerfile defines the runtime environment and application packaging.
- Images are tagged with version identifiers for traceability.
- Validated Docker images are pushed to the **Nexus Repository Manager**.

Role in the CI/CD Pipeline

- Ensures environment consistency across development, testing, and production
- Enables portable and reproducible deployments
- Acts as the deployment unit for Kubernetes

5.4 Nexus Repository – Artifact Management Workflow

Workflow

- Docker images generated by Jenkins are pushed to Nexus.
- Nexus stores and manages build artifacts centrally.
- Versioned artifacts can be reused across environments.

Role in the CI/CD Pipeline

- Centralized and secure artifact storage
- Improves build reliability

- Enables controlled promotion of images

5.5 Kubernetes – Container Orchestration Workflow

Workflow

- Jenkins deploys the validated Docker image to the Kubernetes cluster using `kubectl`.
- Kubernetes pulls the image from Nexus and creates application pods.
- Services expose the application for internal or external access.
- Kubernetes ensures:
  - Automatic pod restarts
  - Load balancing
  - Horizontal scalability

Security Validation

- **kubeaudit** is used to analyze Kubernetes manifests.
- Detects:
  - Privileged containers
  - Missing resource limits
  - Insecure RBAC configurations

Role in the CI/CD Pipeline

- Orchestrates application deployment
- Ensures high availability and fault tolerance
- Enforces runtime security best practices

5.6 SonarQube – Static Code Analysis

Workflow

- SonarQube analyzes the application source code during the CI phase.
- Results are displayed on the SonarQube dashboard.
- Quality gates determine whether the pipeline can proceed.

Role in the CI/CD Pipeline

- Enforces coding standards

- Detects bugs and vulnerabilities early
- Improves maintainability and security

5.7 Trivy – Container and Dependency Security Scanning

Workflow

- Trivy scans:
  - Application file system
  - Docker images before deployment
  - The generated CVE reports are forwarded to n8n, which integrates with the Gemini API to analyze, summarize, and prioritize vulnerabilities based on severity and impact.
  - Identifies known vulnerabilities from public CVE databases.
- Generates detailed vulnerability reports.

Role in the CI/CD Pipeline

- Enhances vulnerability analysis using AI-assisted CVE interpretation
- Reduces manual effort in understanding security scan results
- Improves prioritization of critical security risks

5.8 Prometheus – Monitoring Workflow

Workflow

- Prometheus scrapes metrics from:
  - Kubernetes nodes
  - Pods and containers
  - Application endpoints
- Stores metrics as time-series data.
- Alert rules can be configured for threshold breaches.

Role in the CI/CD Pipeline

- Monitors runtime health and performance
- Enables proactive issue detection

5.9 Grafana – Visualization and Alerting

Workflow

- Grafana queries Prometheus metrics.
- Displays dashboards for:
  - CPU and memory usage
  - Pod health
  - Node availability
- Alerts can be configured for critical conditions.

Role in the CI/CD Pipeline

- Provides real-time observability
- Improves operational visibility
- Supports proactive incident response

5.10 N8N - AI-Assisted Workflow Automation

Workflow

- n8n receives vulnerability scan results and pipeline events from Jenkins through webhook-based integration.
- The CVE data is processed and forwarded to the Gemini API for AI-assisted analysis.
- Gemini generates summarized security insights, severity prioritization, and risk explanations.
- n8n converts the AI output into structured reports and automatically sends notifications to the development and security teams.

Role in the CI/CD Pipeline

- Enables event-driven automation across the CI/CD pipeline
- Implements AI-assisted vulnerability analysis using the Gemini API
- Automates security alerts and reporting
- Improves response time and operational efficiency

IMPLEMENTATION & OUTPUT

Security Groups

Inbound rules (8)

Manage tags | Edit inbound rules

Search

1

Type	Protocol	Port range	Source	Description
HTTP	TCP	80	0.0.0.0/0	–
SMTPS	TCP	465	0.0.0.0/0	–
HTTPS	TCP	443	0.0.0.0/0	–
Custom TCP	TCP	6443	0.0.0.0/0	–
Custom TCP	TCP	3000 - 11000	0.0.0.0/0	–
Custom TCP	TCP	30000 - 32767	0.0.0.0/0	–
SMTP	TCP	25	0.0.0.0/0	–
SSH	TCP	22	0.0.0.0/0	–

JDK Configuration

JDK installations

JDK installations | Edited

Add JDK

JDK

Name

jdk17

Install automatically

Install from adoptium.net

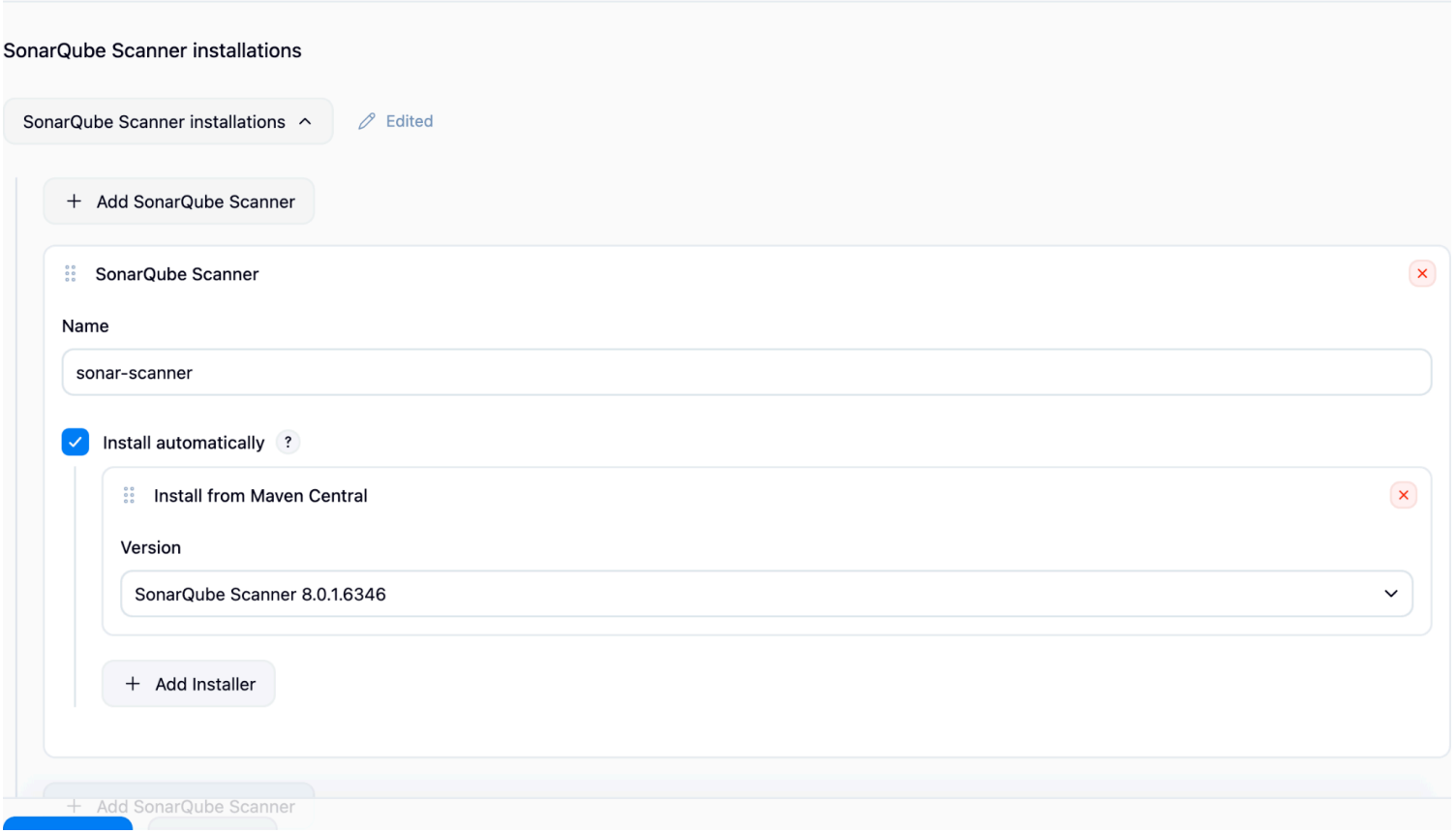
Version

jdk-17.0.18+8

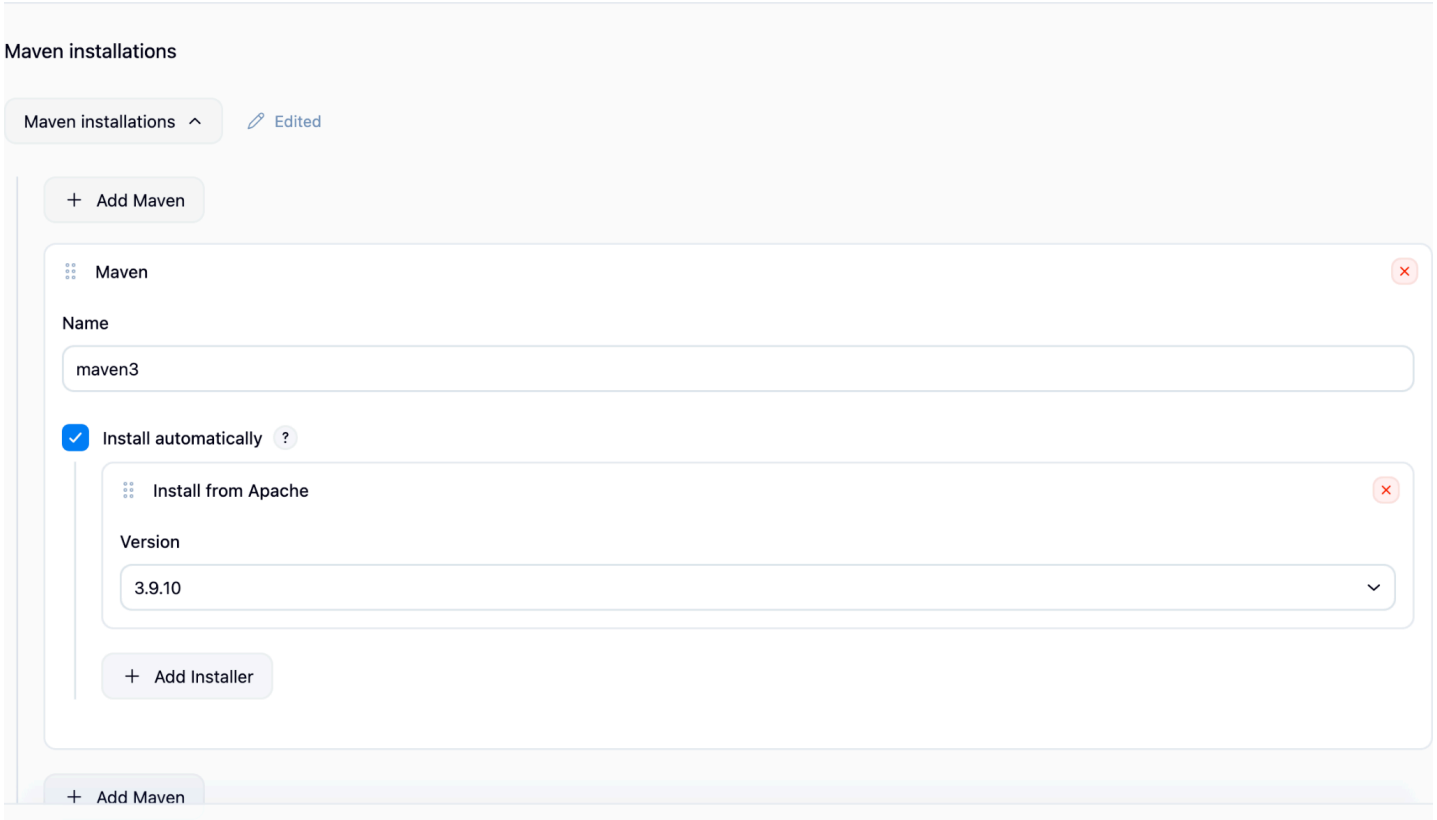
Add Installer

Add JDK

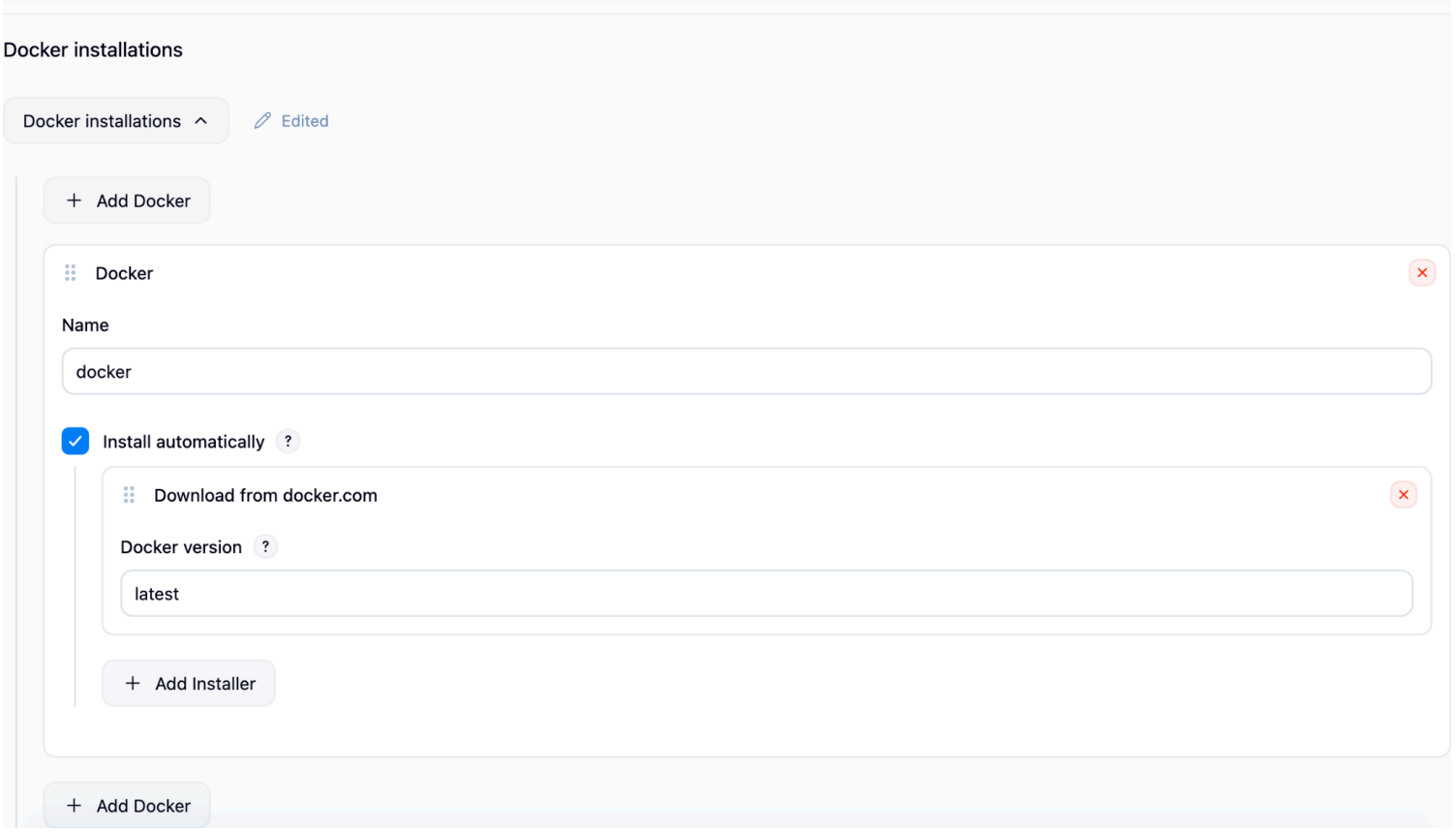
Sonar Scanner Configuration



Maven Configuration

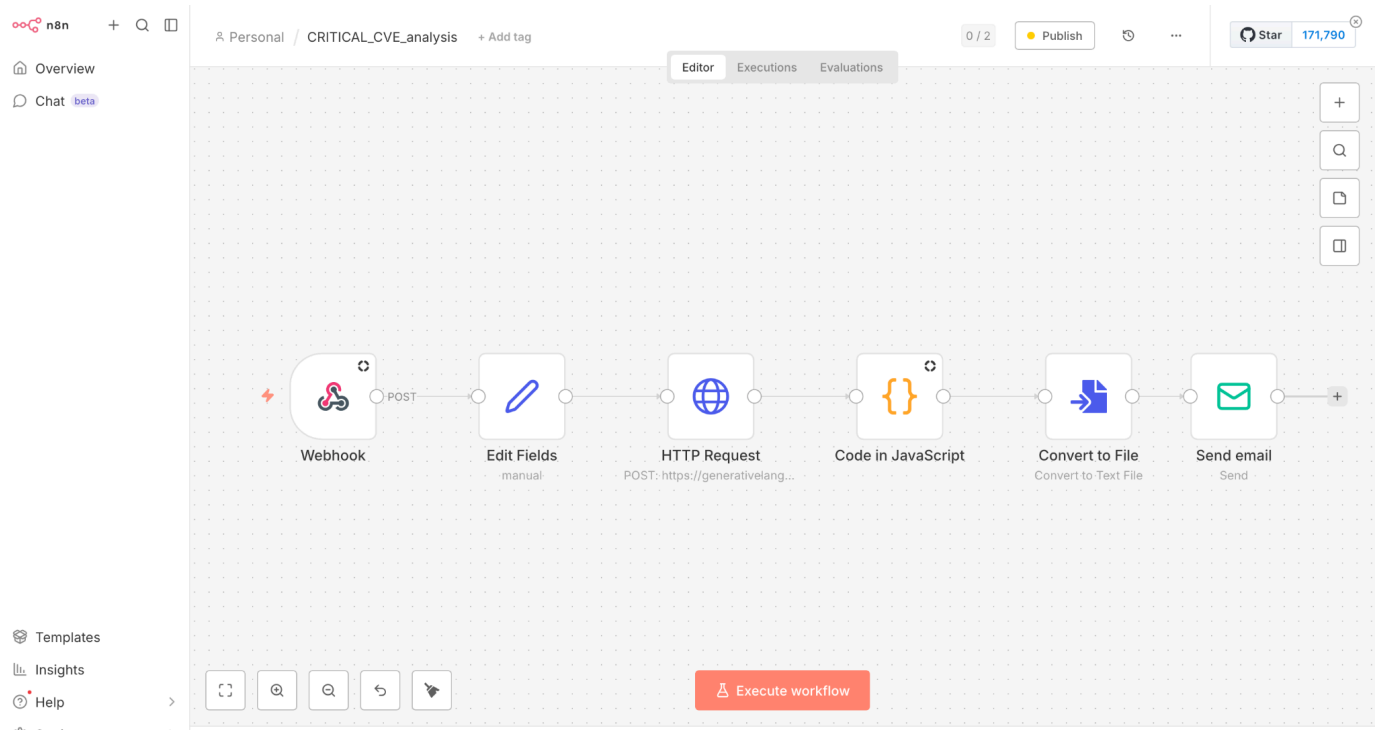


Docker Configuartion

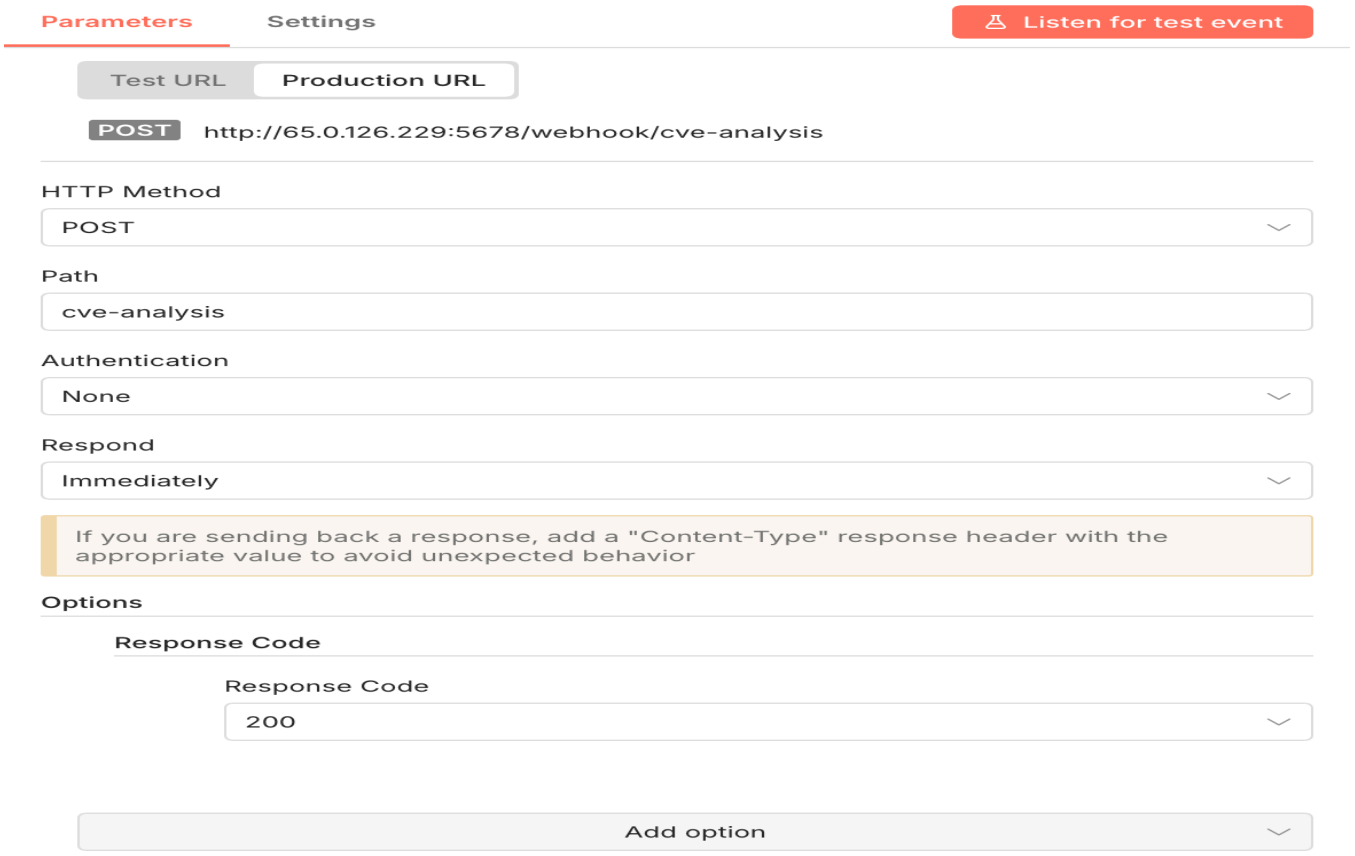


Jenkins

N8N


n8n
Overview
Chat beta
Templates
Insights
Help
Personal / CRITICAL_CVE_analysis + Add tag
0 / 2
Publish
Star 171,790
Editor
Executions
Evaluations
Webhook
POST
Edit Fields
manual
HTTP Request
POST: https://generativelang...
Code in JavaScript
Convert to File
Convert to Text File
Send email
Send
Execute workflow

N8N Webhook


Parameters
Settings
Listen for test event
Test URL
Production URL
POST
http://65.0.126.229:5678/webhook/cve-analysis
HTTP Method
POST
Path
cve-analysis
Authentication
None
Respond
Immediately
If you are sending back a response, add a "Content-Type" response header with the appropriate value to avoid unexpected behavior
Options
Response Code
Response Code
200
Add option

N8N Input Field

Mode

Manual Mapping

Fields to Set

prompt

String

```
You are a cybersecurity expert.\nAnalyze the following CRITICAL CVE
s.\nFor each CVE, provide:\n1. Short description\n2. Why it is dange
rous\n3. Practical remediation steps for DevSecOps / Cloud environme
nts\nCVEs:\n{{ $json.body.replace(/\r?\n/g, "\\n") }}
```

Drag input fields here or Add Field

Include Other Input Fields

Options

No properties

Add option

N8N HTTP Request

Method

POST

URL

https://generativelanguage.googleapis.com/v1beta/models/gemini-2.5-flash:generateConte

Authentication

None

Send Query Parameters

Send Headers

Specify Headers

Using Fields Below

Headers

Name

Content-Type

Value

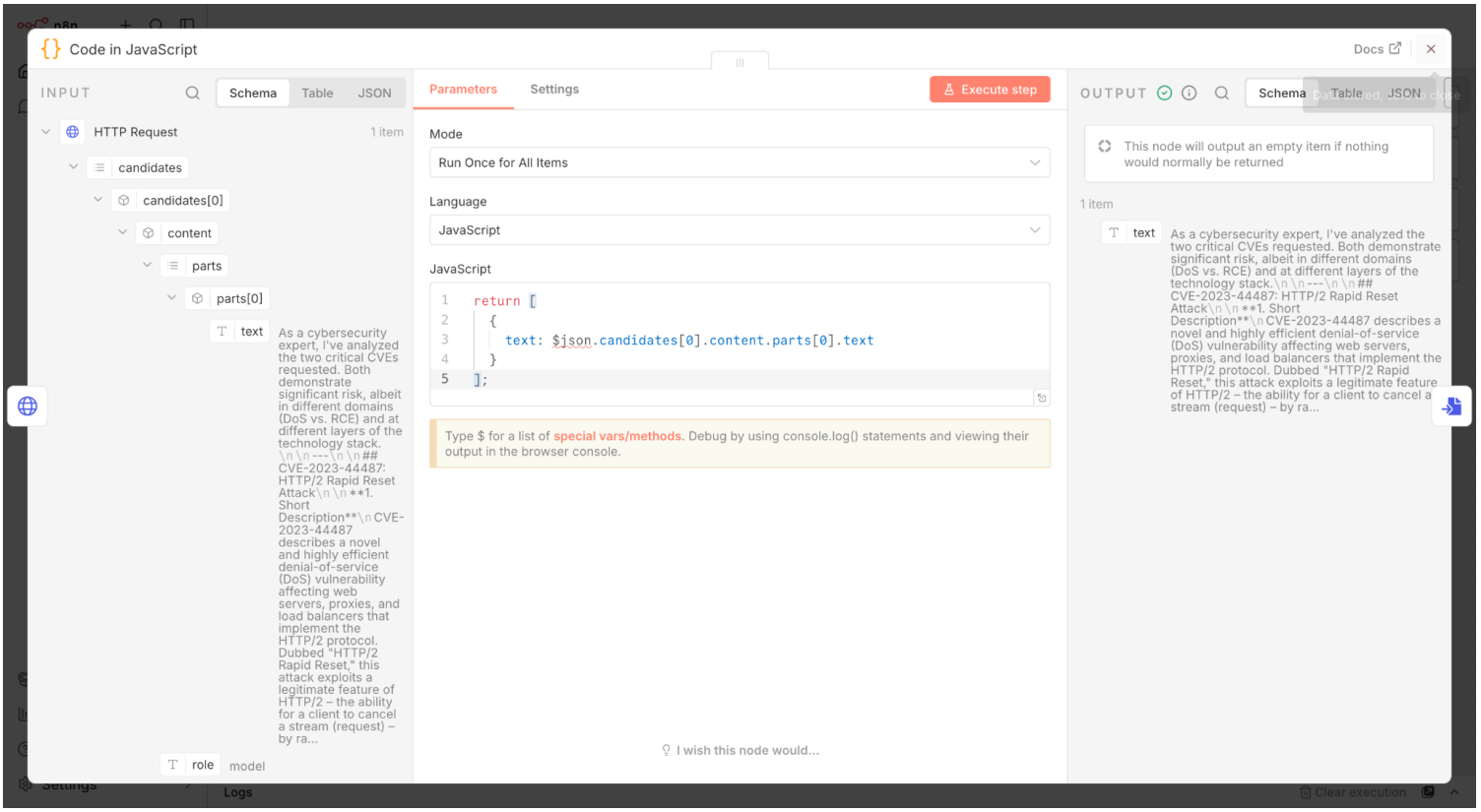
application/json

Add Header

N8N HTTP Requests (1)

N8N HTTP Requests (2)

N8N Code in Javascript



Mode

Run Once for All Items

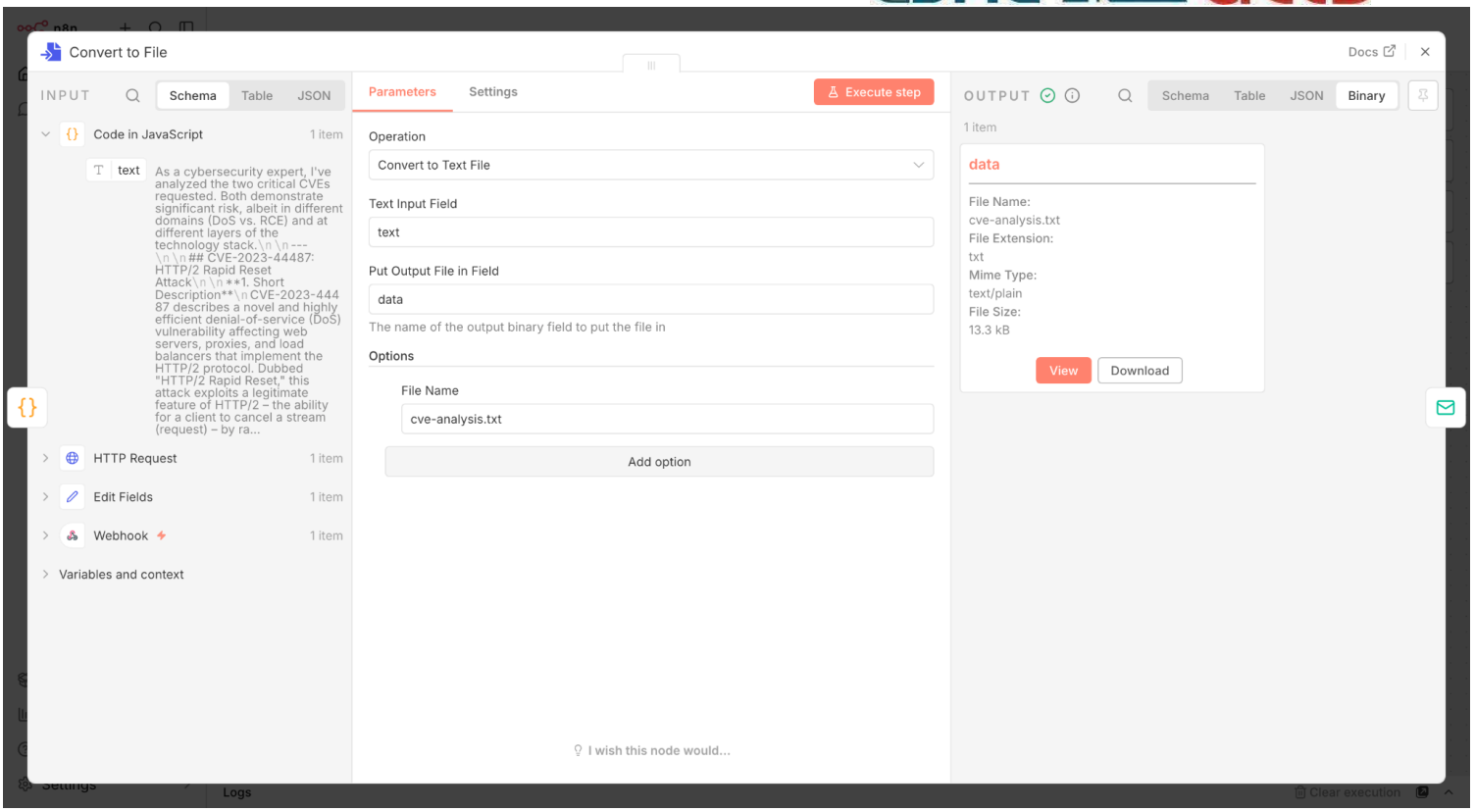
Language

JavaScript

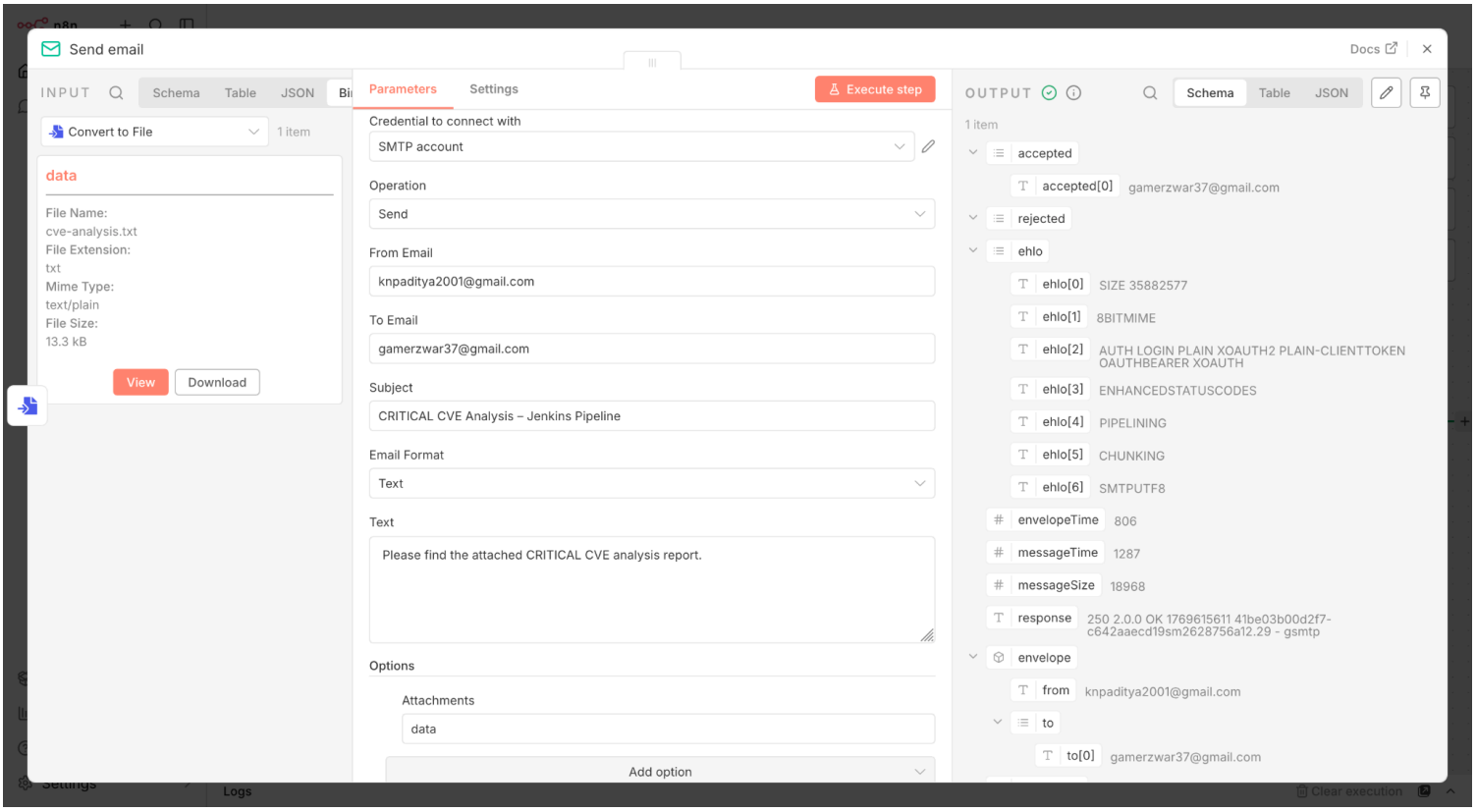
JavaScript

```
return [
  {
    text: $json.candidates[0].content.parts[0].text
  }
];
```

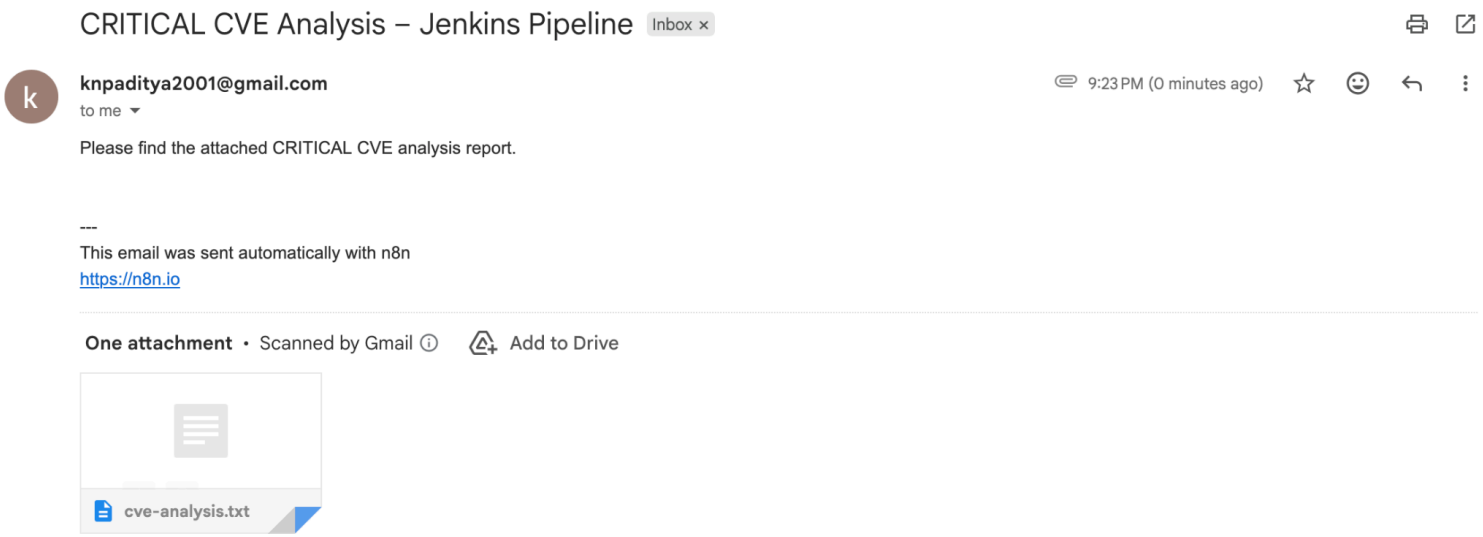
Type $ for a list of **special vars/methods**. Debug by using console.log() statements and viewing their output in the browser console.



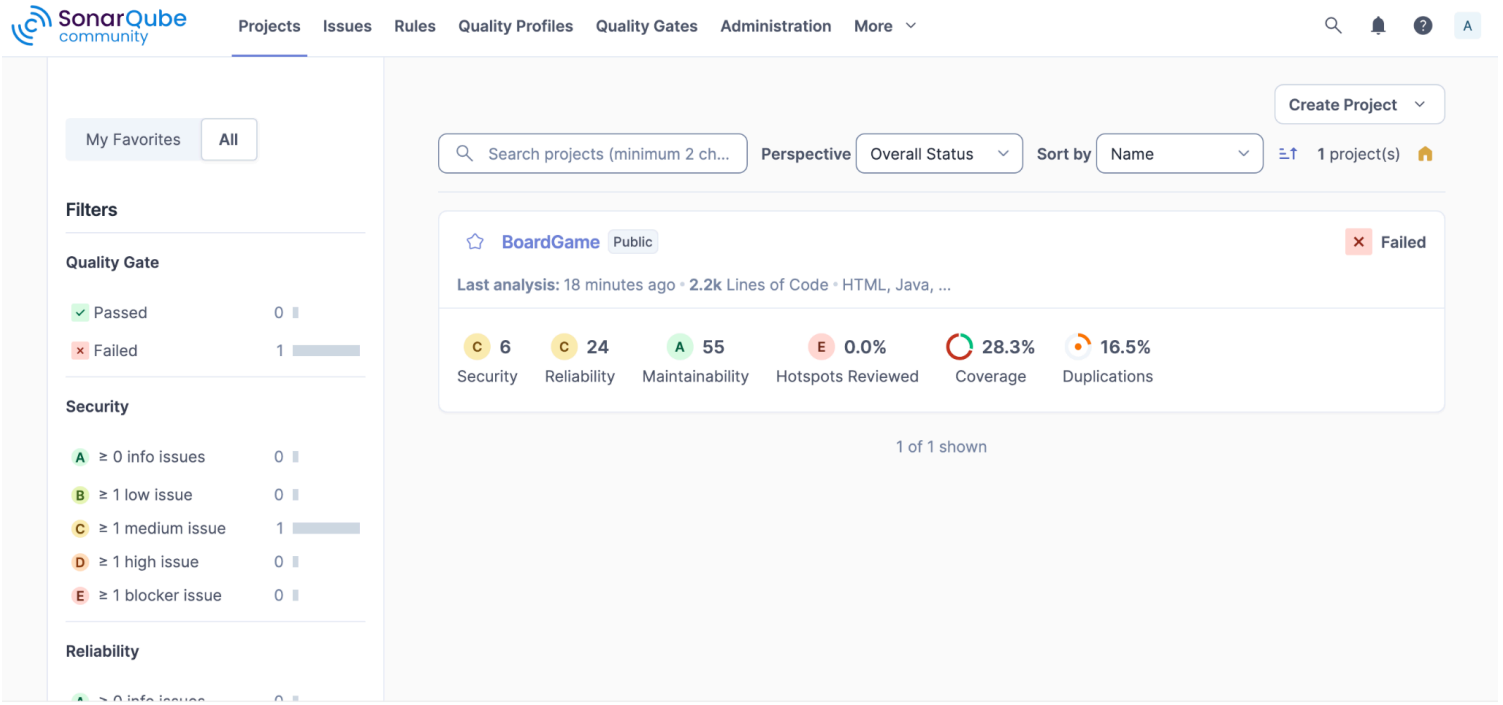
N8N Email Configuration



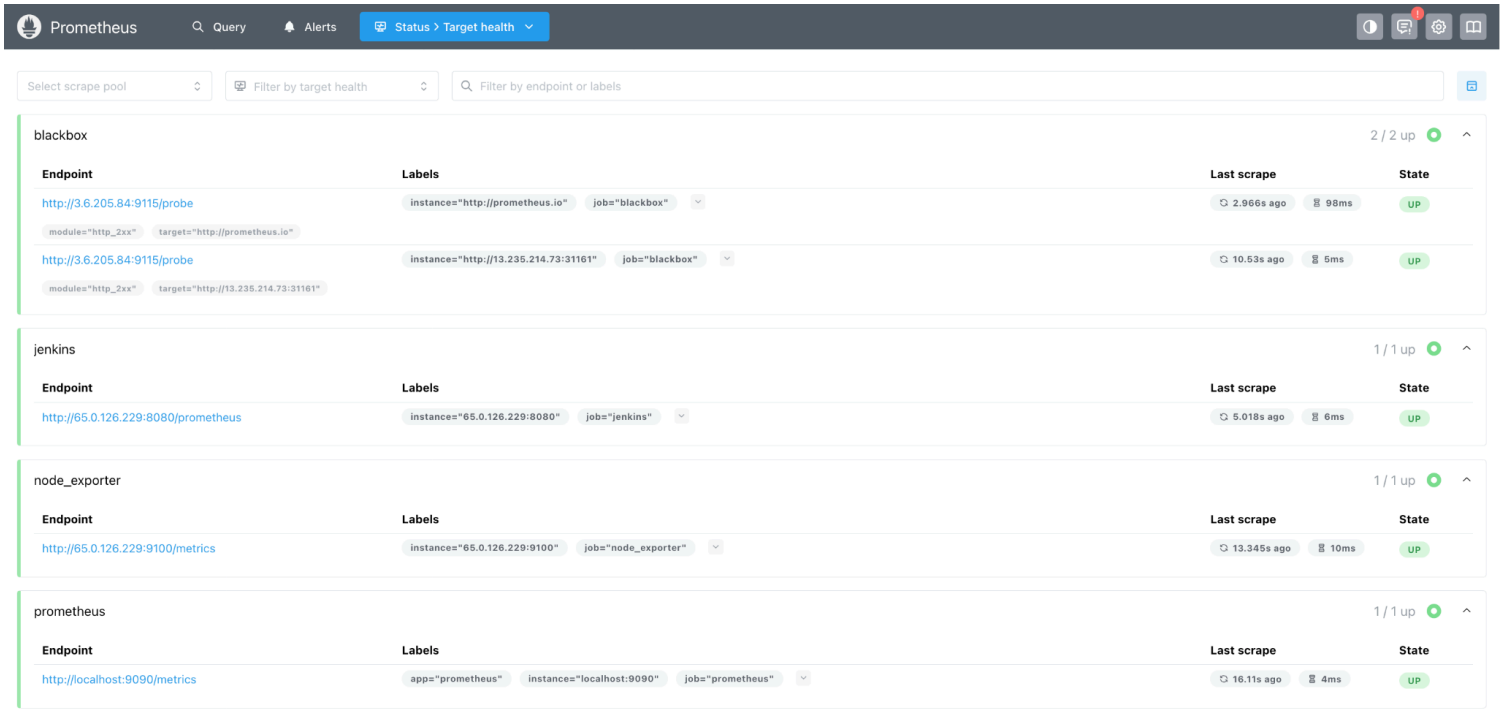
Email Received



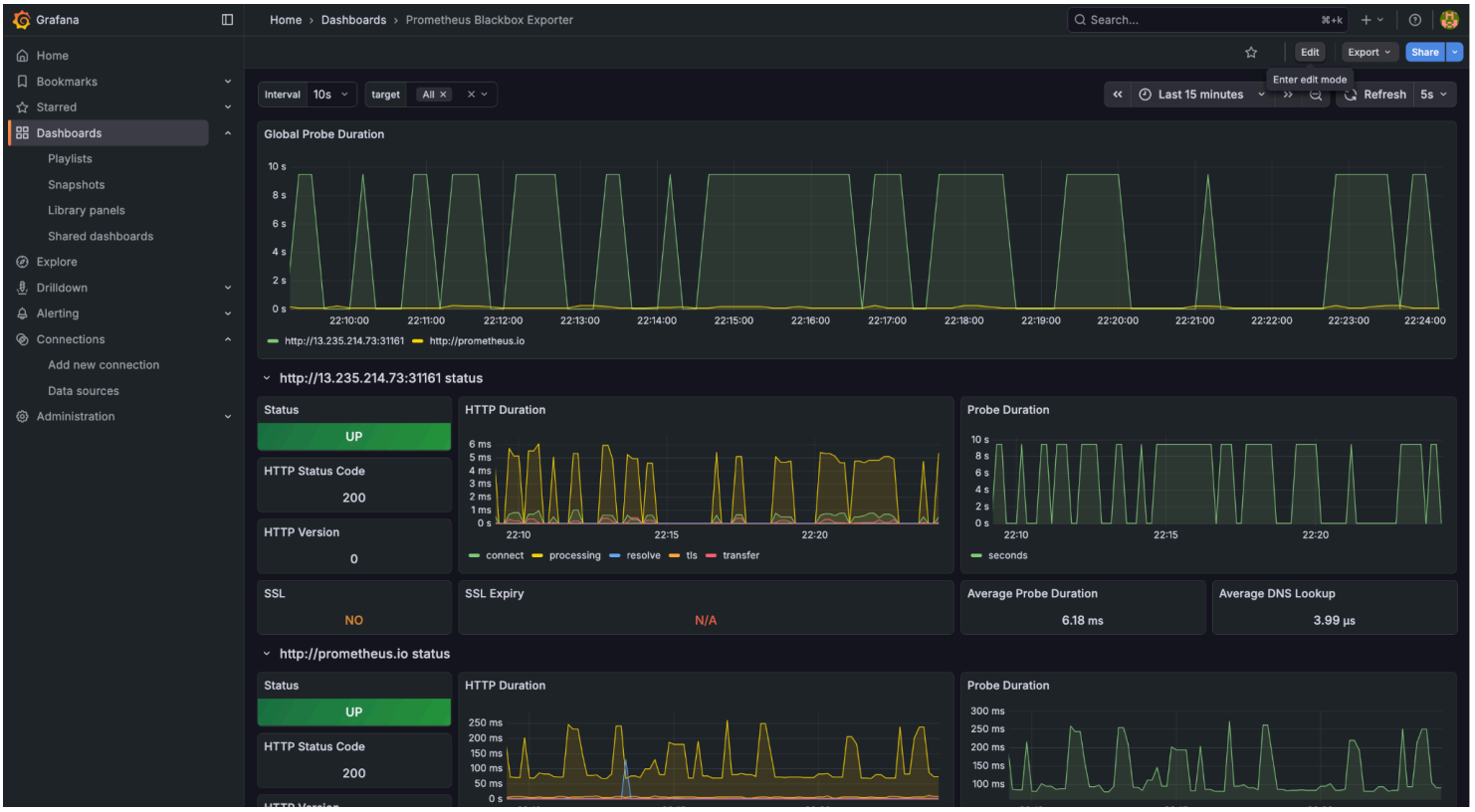
Sonarqube



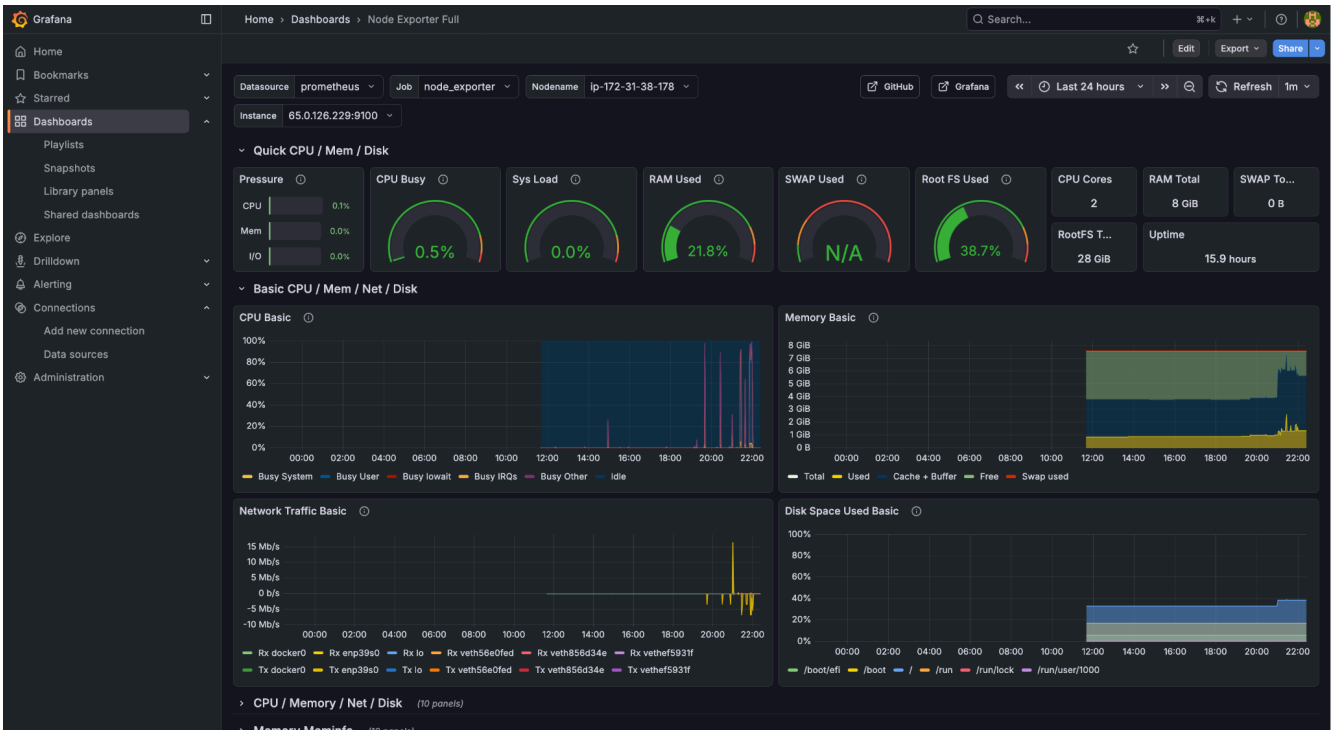
Prometheus



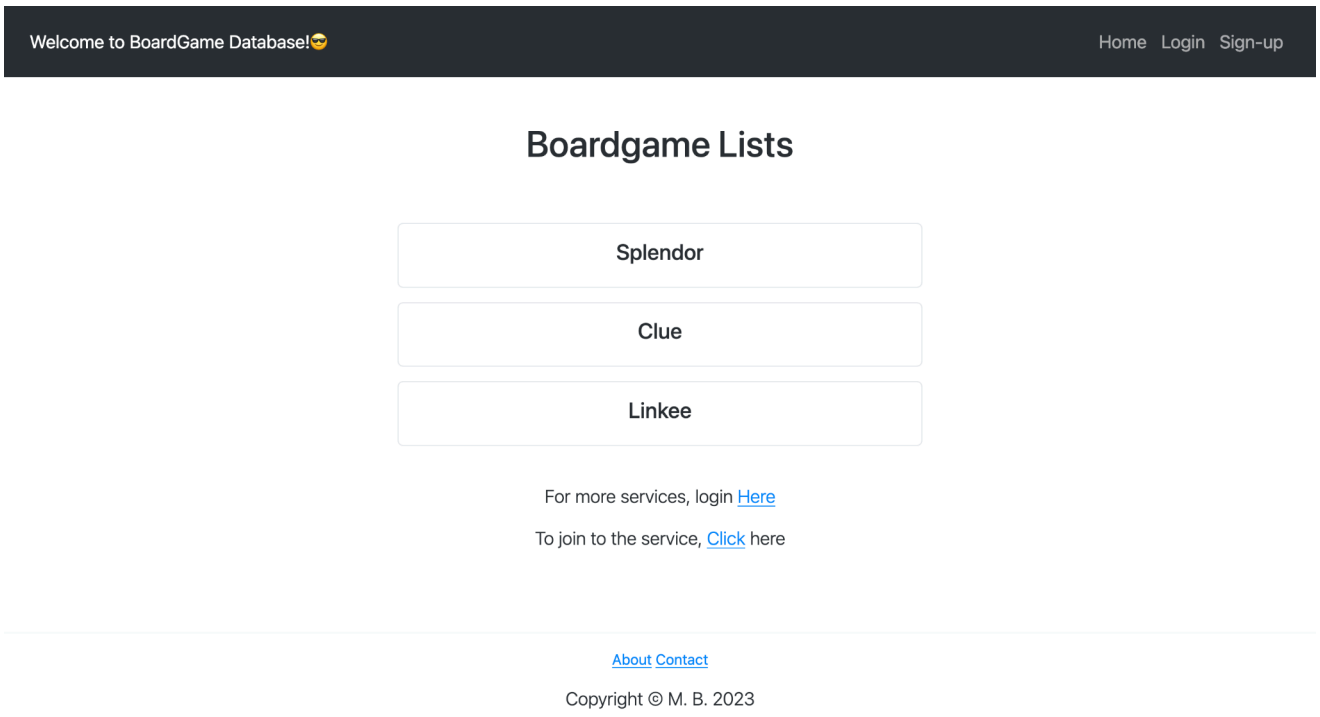
Grafana Dashboard



Grafana Node Exporter Dashboard

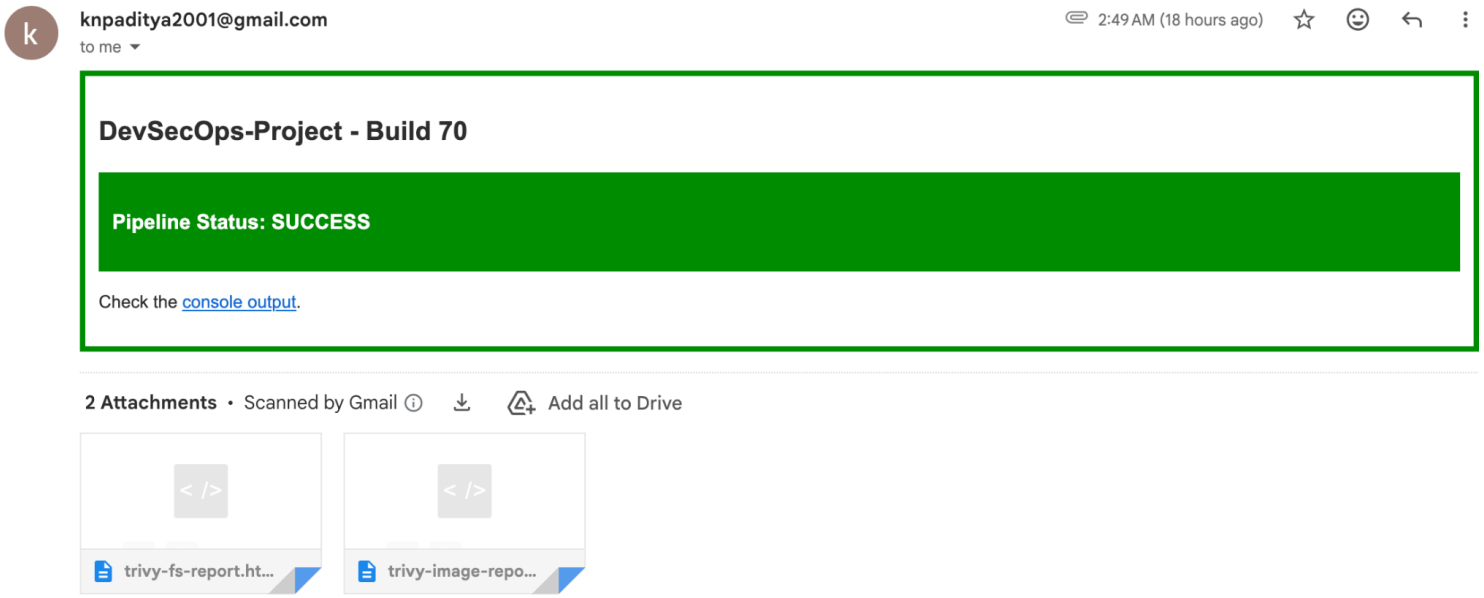


Application



Jenkins CI/CD Pipeline Execution

Email for Pipeline



Kubernetes Pods

```
root@ip-172-31-24-163:/home/ubuntu# kubectl get nodes
NAME               STATUS   ROLES           AGE   VERSION
ip-172-31-18-188   Ready    <none>          16h   v1.35.0
ip-172-31-21-175   Ready    <none>          16h   v1.35.0
ip-172-31-24-163   Ready    control-plane   16h   v1.35.0
root@ip-172-31-24-163:/home/ubuntu# kubectl get pods -n webapps
NAME                                    READY   STATUS    RESTARTS   AGE
boardgame-deployment-5c877fd55c-42mnv   1/1     Running   0          28m
boardgame-deployment-5c877fd55c-bt75c   1/1     Running   0          28m
root@ip-172-31-24-163:/home/ubuntu#
```

CONCLUSION

In this project, we successfully designed and implemented a production-grade DevSecOps CI/CD pipeline using a distributed infrastructure and Kubernetes-based deployment model. The pipeline automates the entire software delivery lifecycle, from source code integration to secure containerized deployment, utilizing Jenkins for CI orchestration, Docker for containerization, and Kubernetes for scalable and reliable application deployment.

By integrating SonarQube for static code analysis and Trivy for container and dependency vulnerability scanning, security and code quality checks were embedded directly into the CI/CD workflow. In addition, kubeaudit was used to validate Kubernetes manifests, ensuring secure cluster configurations prior to deployment. This shift-left security approach ensured that only validated, secure, and high-quality artifacts progressed through the pipeline.

The project further enhances security automation by integrating n8n for event-driven workflow orchestration and the Gemini API for AI-assisted analysis of CVE reports. Vulnerability scan results are intelligently analyzed, summarized, and prioritized using AI, enabling faster and more informed security decision-making while reducing manual effort. Automated notifications and alerts generated through n8n ensure timely response to security and pipeline events.

The use of Nexus Repository Manager enabled centralized and version-controlled artifact storage, improving build reliability and enabling controlled promotion of Docker images across environments. Furthermore, the integration of Prometheus and Grafana provided real-time monitoring and observability of application and infrastructure metrics, allowing proactive detection and resolution of performance or availability issues.

Overall, this project demonstrates how automation, security, AI-assisted analysis, and observability can be effectively combined to build an enterprise-ready DevSecOps CI/CD pipeline. By enforcing security across multiple layers—code, container, Kubernetes, and runtime—the solution aligns with modern industry best practices and provides a scalable, secure, intelligent, and interview-ready foundation for real-world production deployments and future enhancements.

FUTURE SCOPE

The current DevSecOps CI/CD pipeline can be further enhanced by extending its security, scalability, automation, and AI capabilities. While AI-assisted vulnerability analysis is already implemented using the Gemini API, future enhancements can include advanced AI-driven threat correlation, anomaly detection, and predictive risk assessment by leveraging historical vulnerability and monitoring data. Additionally, Kube-bench can be integrated to evaluate Kubernetes cluster security against CIS benchmarks, ensuring continuous compliance with industry security standards.

To improve incident response and operational efficiency, the existing n8n-based automation framework can be expanded to support multi-channel alerting, including SMS, Slack, and incident management tools, alongside email notifications. This would enable faster response to build failures, security vulnerabilities, and runtime incidents. Role-Based Access Control (RBAC) can also be enforced more granularly across Kubernetes resources to ensure strict access control and adherence to the principle of least privilege.

For improved scalability and production readiness, the Kubernetes deployment can be migrated from a self-managed or local cluster to cloud-managed Kubernetes services such as AWS EKS, Google GKE, or Azure AKS. This transition would provide enhanced scalability, fault tolerance, and simplified cluster management. Further enhancements include implementing secure secret management using tools like HashiCorp Vault to protect sensitive credentials, and adopting blue-green or canary deployment strategies with automated rollbacks to ensure zero-downtime releases and faster recovery from failures.

These future enhancements would elevate the pipeline into a fully autonomous, cloud-native, and AI-augmented DevSecOps platform, capable of supporting enterprise-scale workloads while maintaining strong security, reliability, and operational excellence.

REFERENCES

1. GitHub Documentation. *GitHub – Version Control and Collaboration Platform.* Available at: https://docs.github.com/en
2. Jenkins Documentation. *Jenkins Continuous Integration and Automation Server.* Available at: https://www.jenkins.io/doc/
3. Docker Documentation. *Docker and Docker Hub – Containerization Platform.* Available at: https://docs.docker.com/docker-hub
4. SonarSource. *SonarQube Server Documentation (Version 9.9).* Available at: https://docs.sonarsource.com/sonarqube-server/9.9/
5. Aqua Security. *Trivy – Vulnerability Scanner for Containers and Filesystems.* Available at: https://aquasecurity.github.io/trivy/
6. Kubernetes Documentation. *Kubernetes – Container Orchestration Platform.* Available at: https://kubernetes.io/docs/
7. Sonatype. *Nexus Repository Manager Documentation.* Available at: https://help.sonatype.com/repomanager3
8. Prometheus Authors. *Prometheus Monitoring and Alerting Toolkit.* Available at: https://prometheus.io/docs/
9. Grafana Labs. *Grafana Visualization and Monitoring Platform.* Available at: https://grafana.com/docs/
10. Aqua Security. *kubeaudit – Kubernetes Security Auditor.* Available at: https://github.com/Shopify/kubeaudit
11. n8n Documentation. *n8n – Workflow Automation and Integration Platform.* Available at: https://docs.n8n.io/
12. Google AI Documentation. *Gemini API – AI Models and Developer Documentation.* Available at: https://ai.google.dev/