**Project Report**

**University of Petroleum and Energy Studies (UPES)**

**Topic**

FinSureTech: DevSecOps Pipeline for Financial Web App

Under the Guidance of (Mentor): **Utkarsh Agarwal**

**Team Members**:

Rajat Kamboj (500105058)

Raghav Kamboj (500102126)

Aayush Mishra (500

Pulkit Kumar (500107576)

Ansh Saxena (500108581)

Kushagra Singh (500

**Problem Statement:** FinSureTech: DevSecOps Pipeline for Financial Web App

**Project Overview**

FinSureTech, a fintech startup developing a customer-facing financial web application, must  
meet strict regulatory and security compliance requirements. Their current delivery pipeline  
lacks integrated security checks, leading to late detection of vulnerabilities and delayed  
releases—posing risks to both customer trust and audit readiness. To address this,  
FinSureTech needs a secure CI/CD pipeline that embeds security at every stage of the  
development lifecycle. This includes early detection of vulnerabilities in both source code  
and open-source dependencies, runtime security analysis to catch real-time issues, secure  
handling of sensitive credentials, and consistent deployment environments—all without  
slowing down delivery. The company seeks a solution that automates these processes,  
ensuring rapid and secure delivery of features while staying compliant with industry  
standards.

**Key Features**CI/CD with Security Gates: Configure a Jenkins pipeline with stages for Static  
Application Security Testing (SAST) using SonarQube and Software Composition Analysis (SCA) with Snyk.  
Container Scanning: Scan Docker images with Clair or Trivy to identify vulnerabilities in the operating system or libraries.  
 Dynamic Analysis: Use OWASP ZAP for Dynamic Application Security Testing (DAST) in a staging environment to detect runtime issues.  
Infrastructure Provisioning: Provision a secure AWS environment with Terraform for staging and production deployments.  
Secrets Management: Securely manage sensitive data (e.g., API keys) using AWS Secrets Manager or Kubernetes secrets.

**Tech Stack:**GitHub: Version control  
Docker: Containerization  
Jenkins: CI/CD automation  
Terraform: Infrastructure provisioning  
SonarQube: Static analysis  
OWASP ZAP: Dynamic scanning  
EC2/ECS: Deployment platform  
Synk/Clair: Additional security tools

**Objective:**

To create a fully automated, secure CI/CD pipeline for a Node.js financial application deployed on AWS EC2. The pipeline integrates security scanning tools (SAST, SCA, Image Scan, DAST), automatic deployments, and AI-driven monitoring.

**Key Features:**

* Fully automated Jenkins pipeline
* Static code analysis with **SonarQube**
* Software composition analysis with **Snyk**
* Container image scanning using **Trivy**
* Deployment to **EC2 (Staging & Production)** via Docker
* Dynamic security testing with **OWASP ZAP**
* Monitoring with **Prometheus + Grafana**
* Smart Alerts and AI-based **anomaly detection** in Grafana

**Tools Used:**

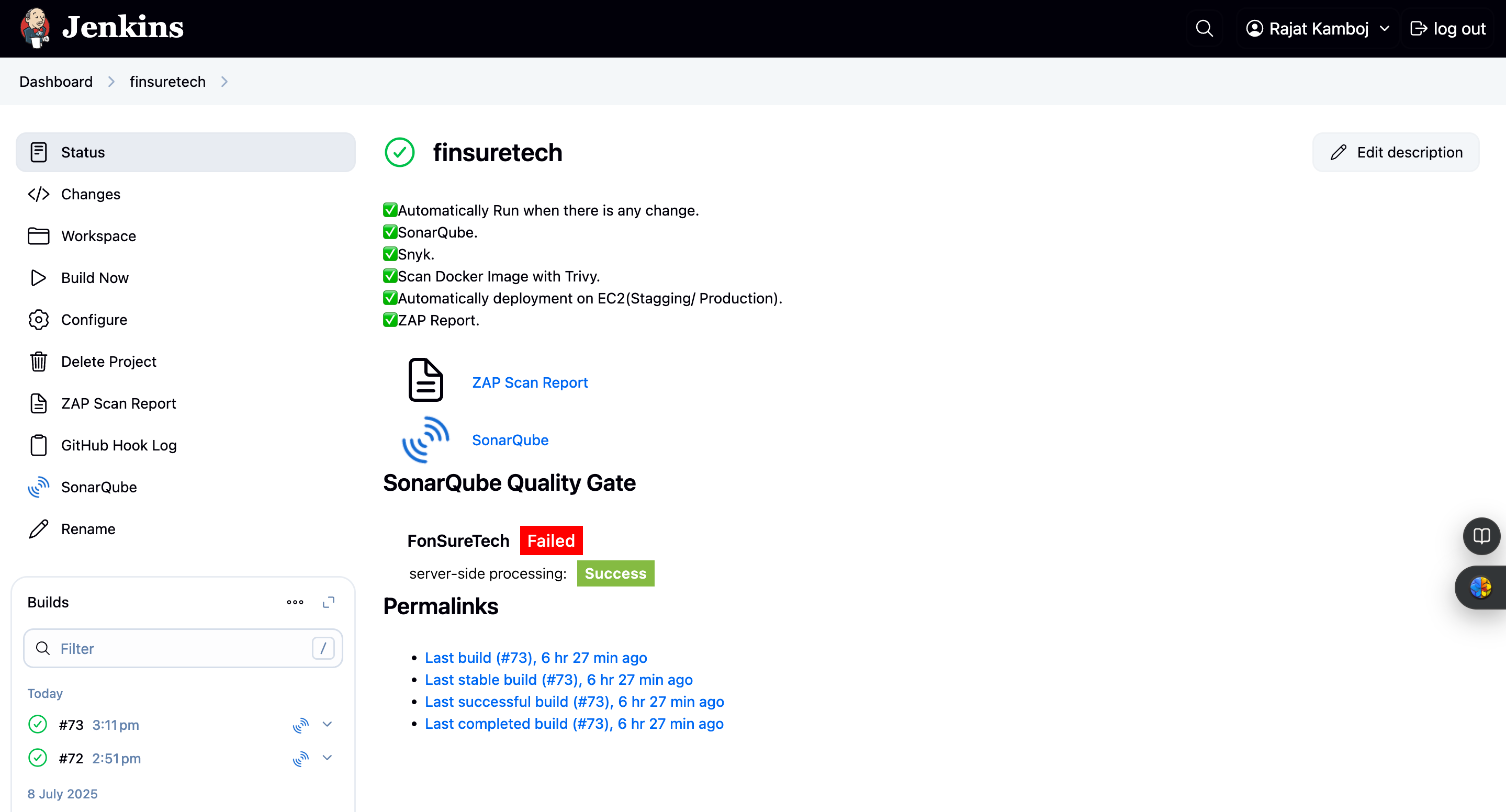
|  |  |
| --- | --- |
| Tool | Purpose |
| Jenkins | CI/CD orchestration |
| SonarQube | Static Analysis (SAST) |
| Snyk | Dependency Vulnerability Check (SCA) |
| Docker | Containerization |
| Trivy | Docker Image Scanning |
| AWS EC2 | Application Hosting |
| OWASP ZAP | DAST Security Scan |
| Prometheus | Metrics Collection |
| Grafana | Monitoring & Anomaly Detection |
| GitHub | Version Control |

**Jenkins – CI/CD Automation Engine:**

Jenkins is an open-source automation server that enables **Continuous Integration (CI)** and **Continuous Delivery (CD)**. It automates building, testing, and deploying software whenever code changes are made, ensuring fast and reliable delivery.

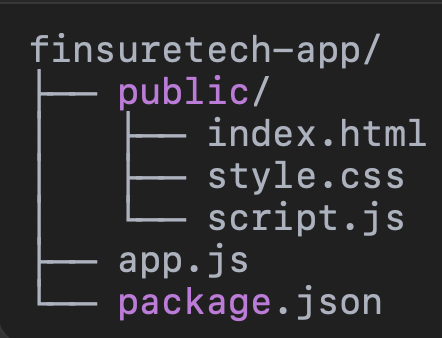
**Role of Jenkins in FinSureTech:** In the FinSureTech DevSecOps pipeline, Jenkins acts as the **orchestrator**—automating every step of the development-to-deployment.

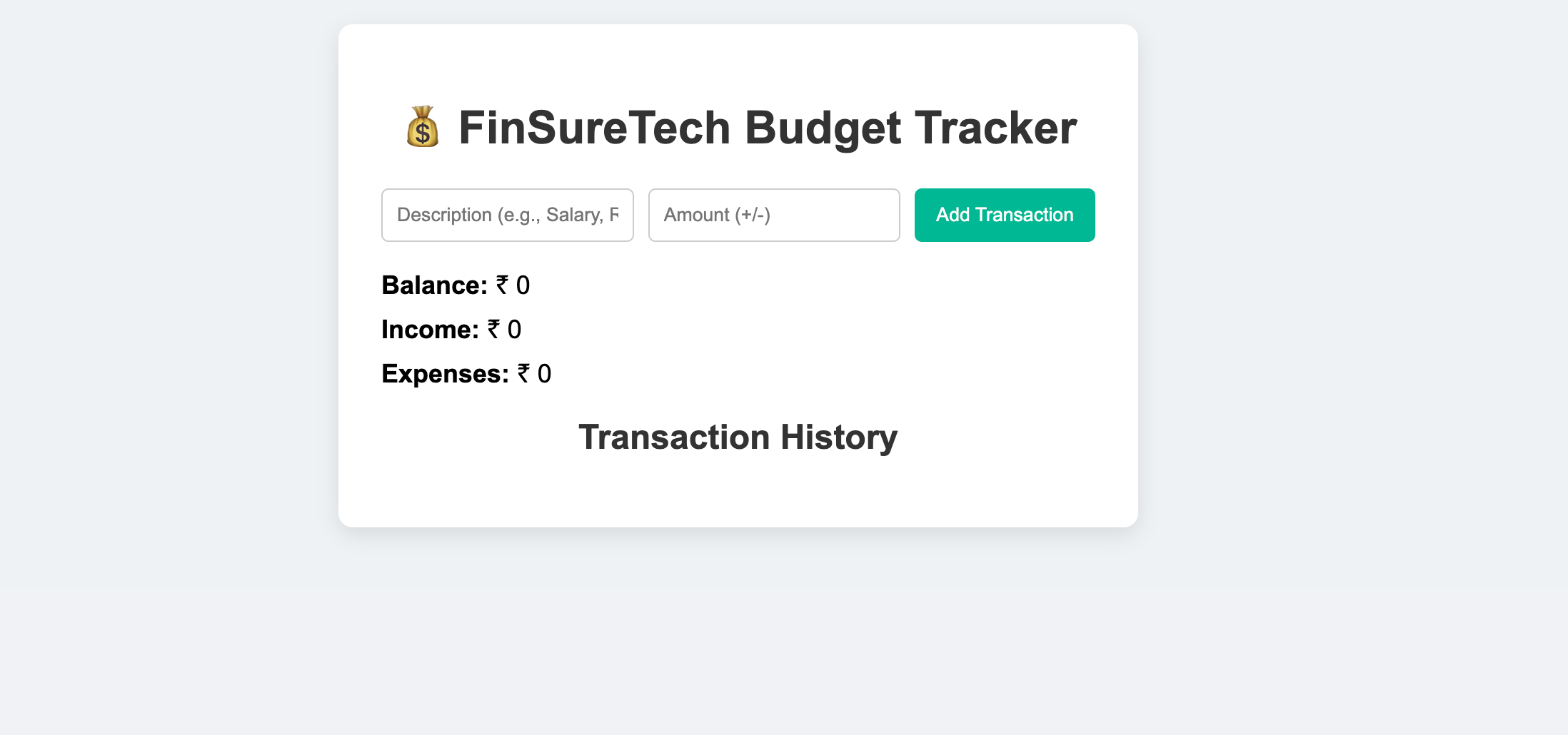
**Jenkins Work as :** Automatically Run when there is any change.  
SonarQube.  
Snyk.  
Scan Docker Image with Trivy.  
Automatically deployment on EC2(Stagging/ Production).  
ZAP Report.

****

**FinSureTech Frontend:**

The **FinSureTech frontend** is a lightweight, user-friendly web interface designed to simulate a financial dashboard or service portal. Built using **HTML, CSS, and JavaScript**, it serves as the visual component of the full-stack application, providing users with access to key financial features.

****

****

**Automatic Trigger on Code Change  
Tool:** Jenkins (Git webhook or polling)

Automatic triggering refers to the ability of Jenkins to **detect a new code push** (e.g., via GitHub) and **immediately start the CI/CD pipeline**, without manual intervention.

**1. GitHub Webhook Integration:** A **webhook** is configured in your GitHub repository.  
Whenever a developer pushes code or creates a pull request (PR), GitHub sends an HTTP POST request to Jenkins.

**2. Jenkins GitHub Plugin:** Jenkins must have the **“GitHub Integration Plugin”** or **“GitHub webhook trigger for GITScm polling”** installed.

In your Jenkins job configuration:Check **“GitHub hook trigger for GITScm polling”**

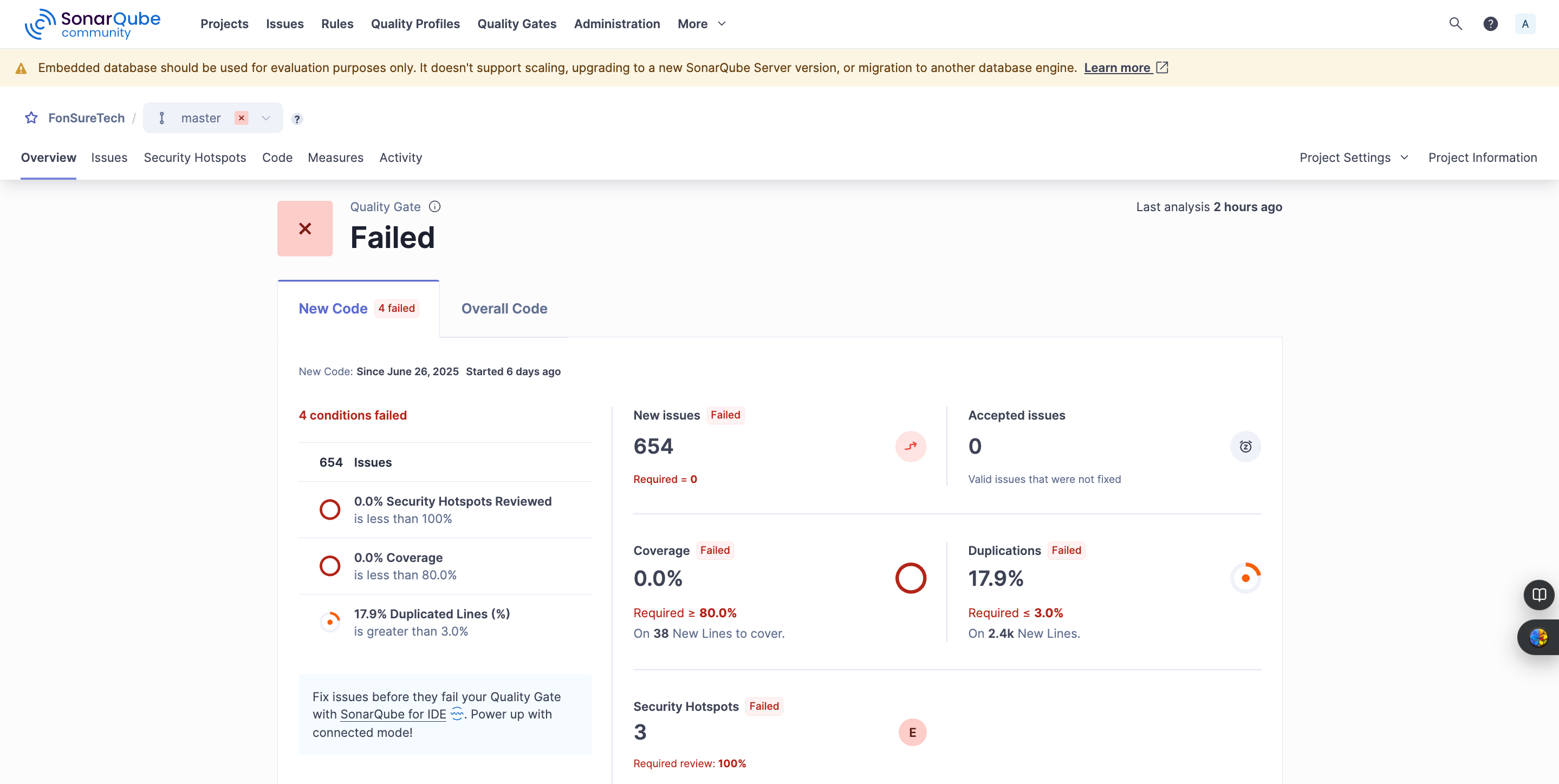
**SonarQube (SAST - Static Application Security Testing)**

**Purpose:** Finds bugs, code smells, and security vulnerabilities in your code.

**SonarQube** is an **open-source platform** used to **inspect the code quality, maintainability, reliability, and security** of software applications.

In your DevSecOps pipeline, it performs **Static Application Security Testing (SAST)** — meaning it scans the source code **without running the application** to detect potential **bugs, code smells, and vulnerabilities**.

**Result:** Dashboards show issues categorized by severity (Critical, Major, Minor).



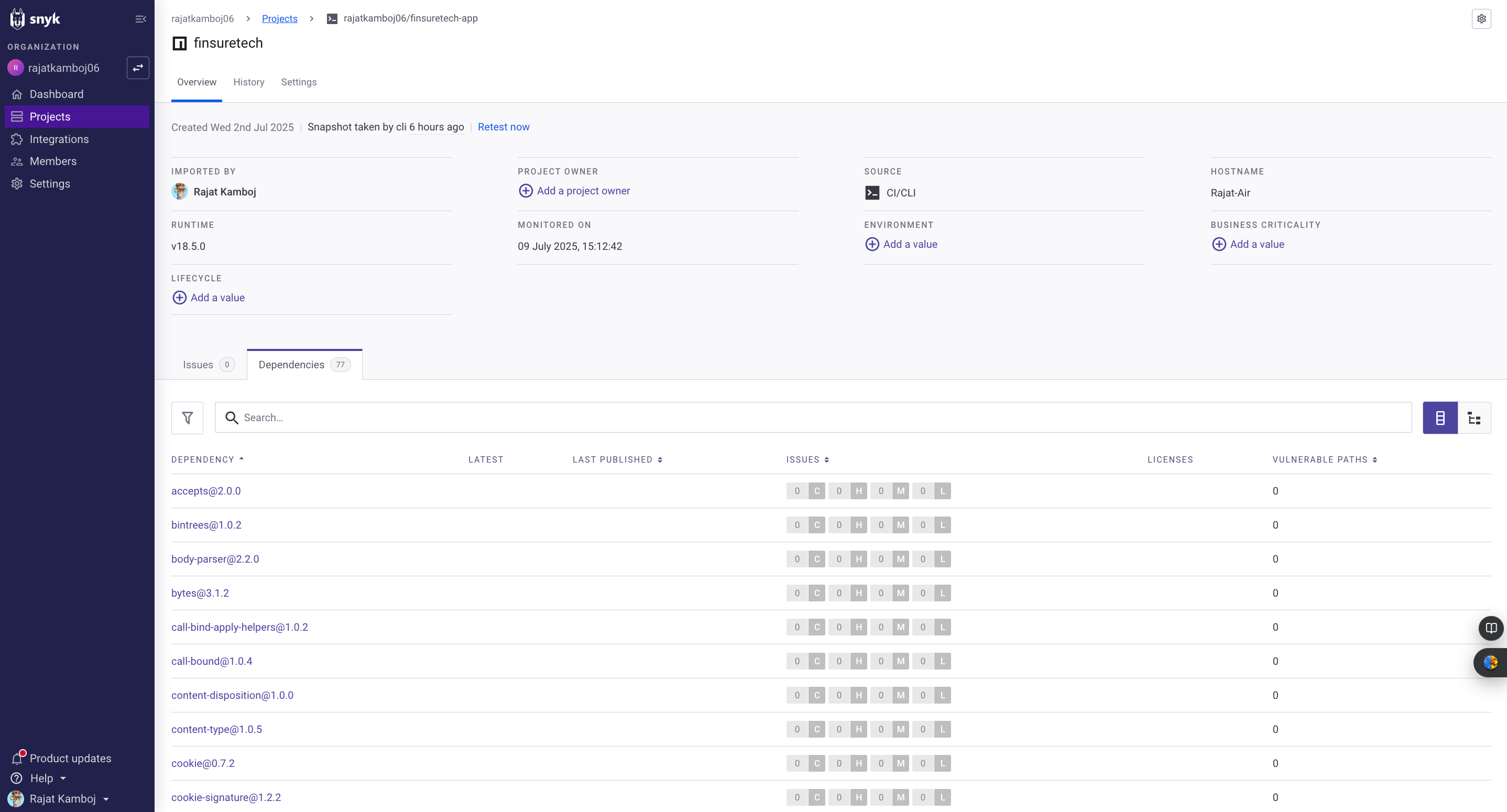
**Snyk (SCA - Software Composition Analysis)**

**Purpose:** Scans package.json (Node.js) for open-source vulnerabilities.

**Snyk** is a developer-friendly tool that performs **Software Composition Analysis (SCA)**. It scans your project’s dependencies (like NPM packages, Docker base images, etc.) to identify **known vulnerabilities** and **license risks** in open-source components.

In the FinSureTech DevSecOps pipeline, Snyk is integrated to ensure that third-party libraries do not introduce security flaws.

**Result:** Identifies vulnerable dependencies with fix suggestions.



**Trivy (Docker Image Scanning)**

**Purpose:** Scans built Docker images for OS-level vulnerabilities.

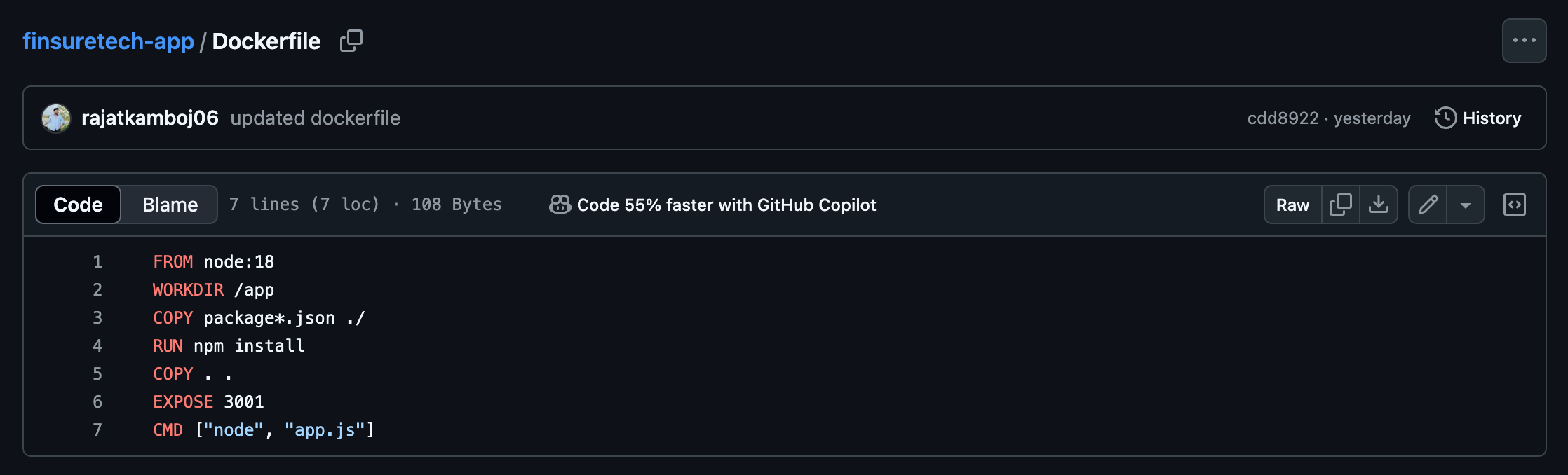
**Trivy** is a comprehensive **open-source vulnerability scanner** for Docker images, File systems, Git repositories, Kubernetes manifests, IaC (Infrastructure as Code) like Terraform

In the **FinSureTech** pipeline, Trivy is used to **scan Docker images** for:

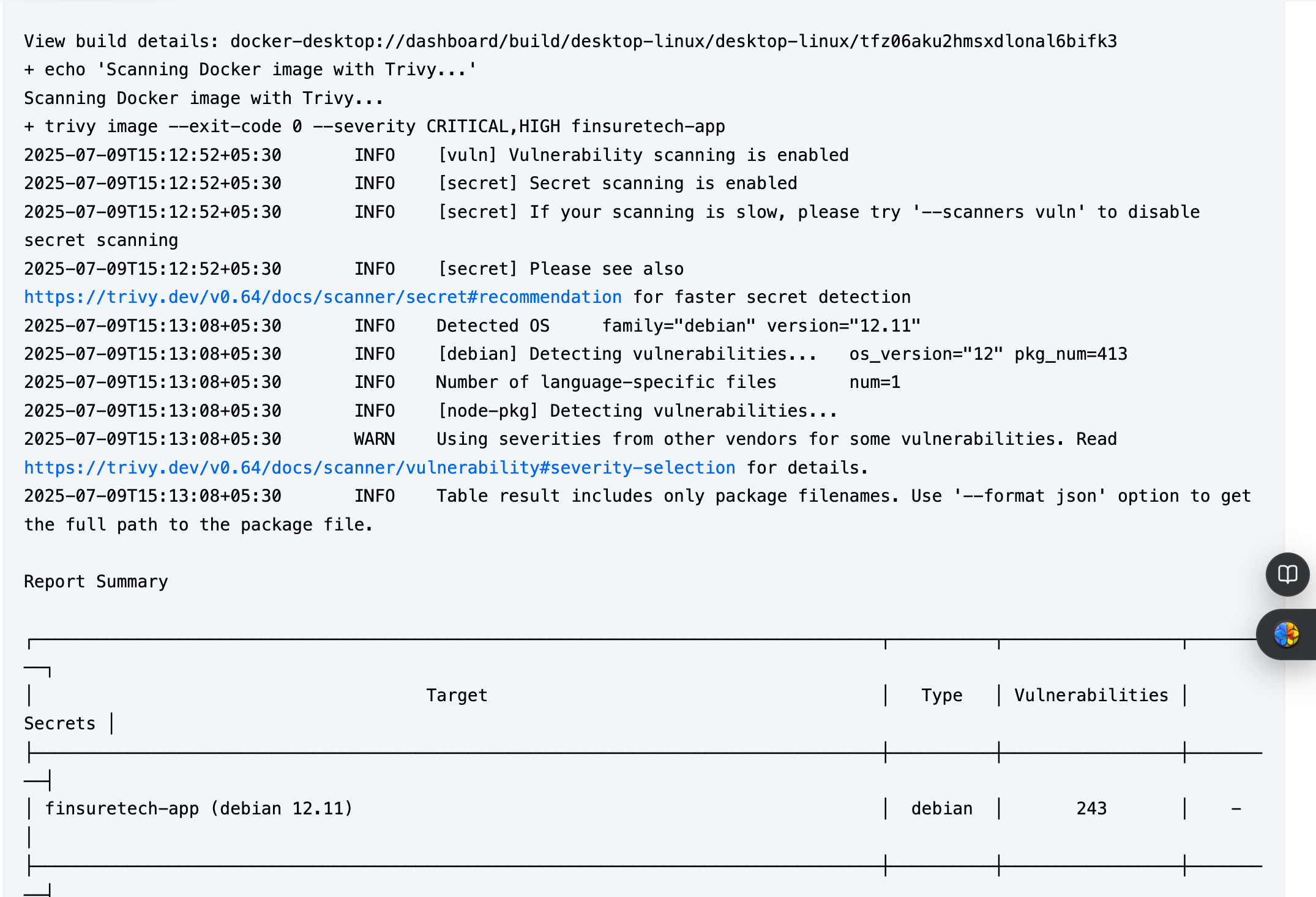
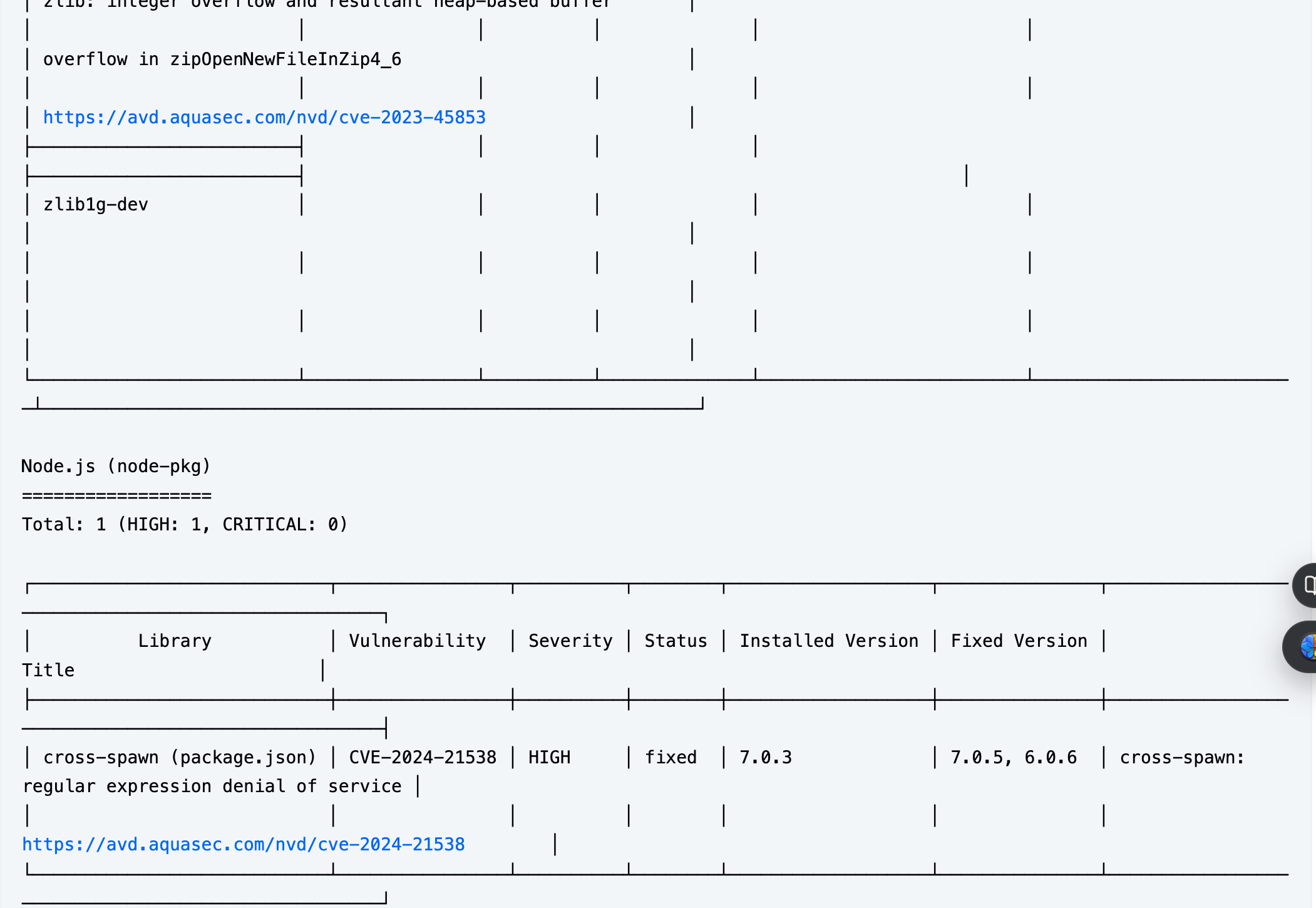
* **OS package vulnerabilities** (e.g., Alpine, Ubuntu, etc.)
* **Application library vulnerabilities** (e.g., NPM, Python, Go)

**Result:** Report with CVEs, severity, and fix info.

DockerFile:



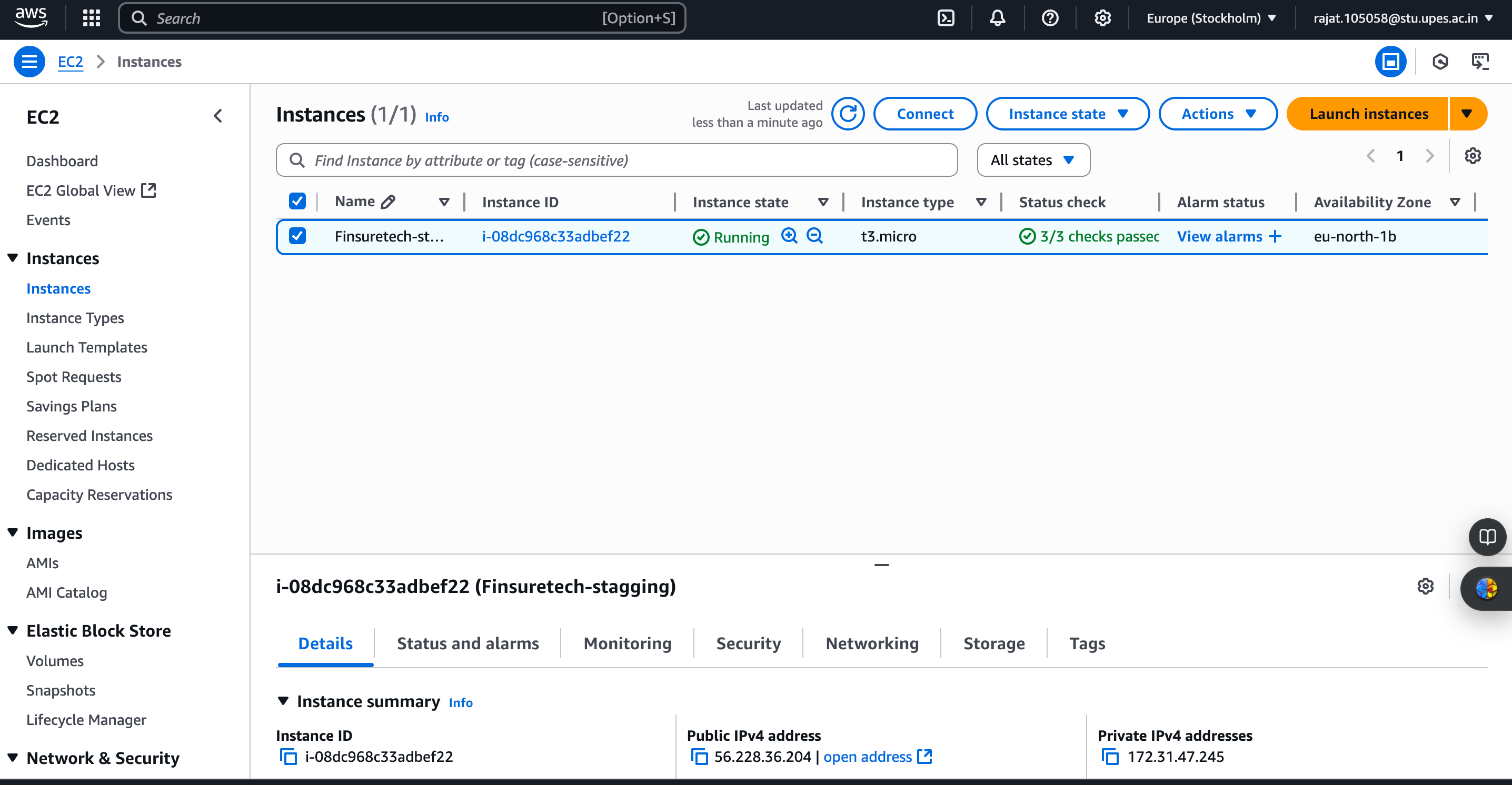
Trivy output:

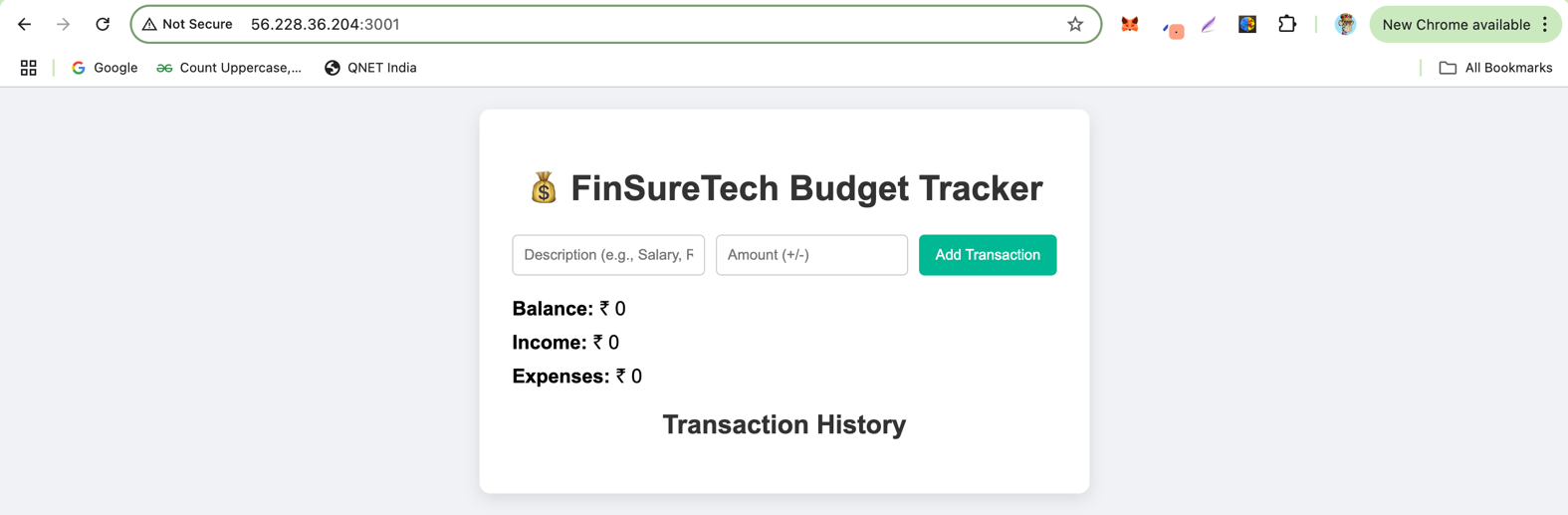
  
  


**Deployment on EC2: Staging / Production EC2** via Docker:

**Amazon EC2 (Elastic Compute Cloud)** provides virtual servers (instances) to host applications.  
In **FinSureTech**, two EC2 environments are used:

* **Staging EC2** – for testing and validation.
* **Production EC2** – for live, user-facing deployment.

**EC2 Instance**:  


Hosted at: <http://56.228.36.204:3001>  


**OWASP ZAP (DAST):**

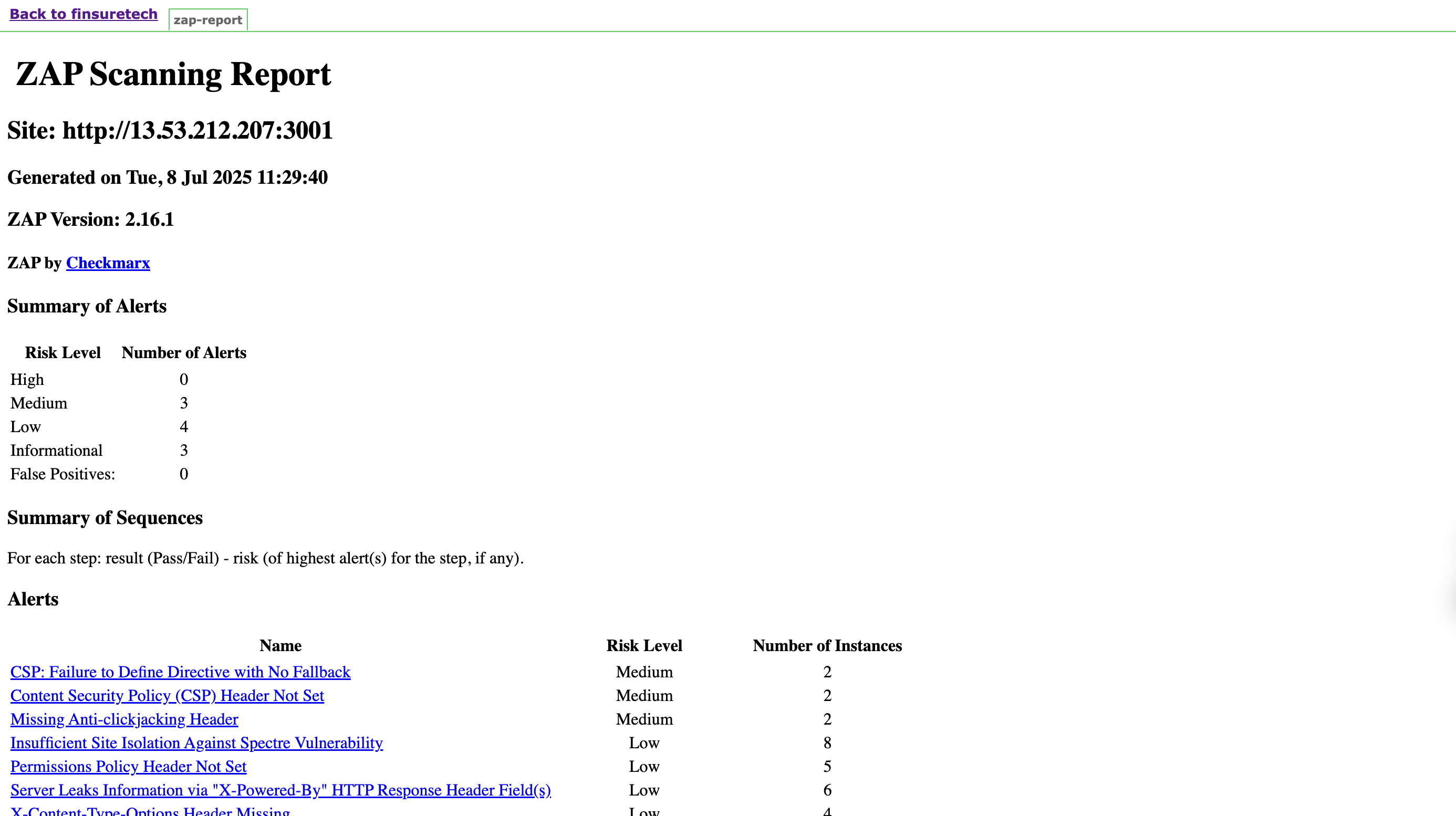
**Purpose**: Automatically scan the running app after deployment.

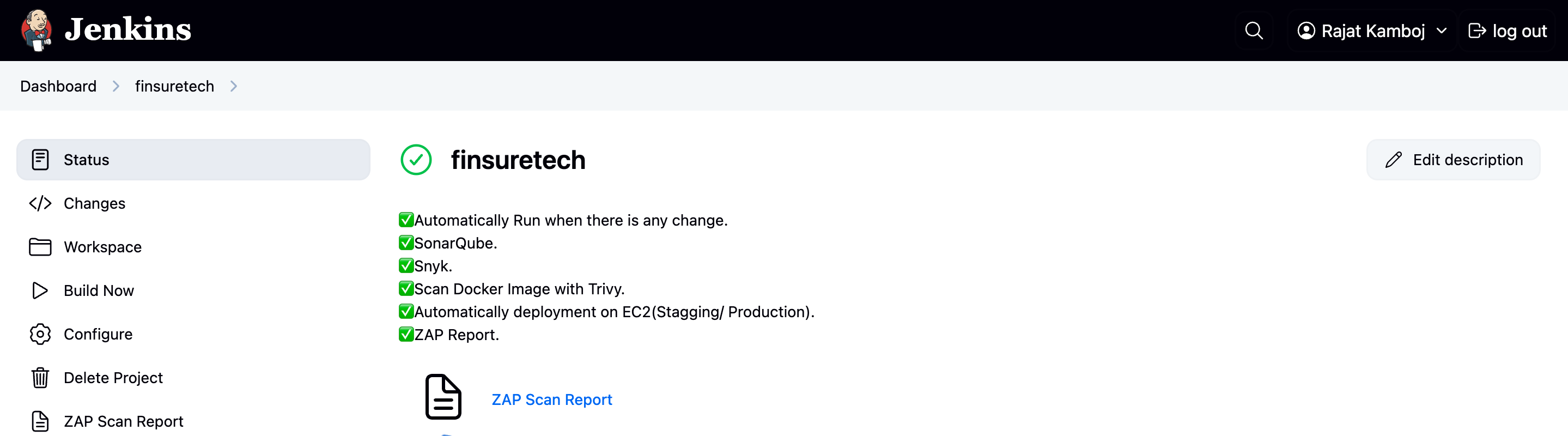
**OWASP ZAP** (Zed Attack Proxy) is a **DAST tool** that scans the **live application** for vulnerabilities **after deployment**.

In **FinSureTech**, it is used to:

* Perform **automated security scanning** of the running Node.js app.
* Find issues like missing headers, cookie weaknesses, outdated libraries, and more.
* Generate an **HTML vulnerability report** for Jenkins and Grafana dashboards.

**ZAP report at:** <http://localhost:8080/job/finsuretech/ZAP_20Scan_20Report/>



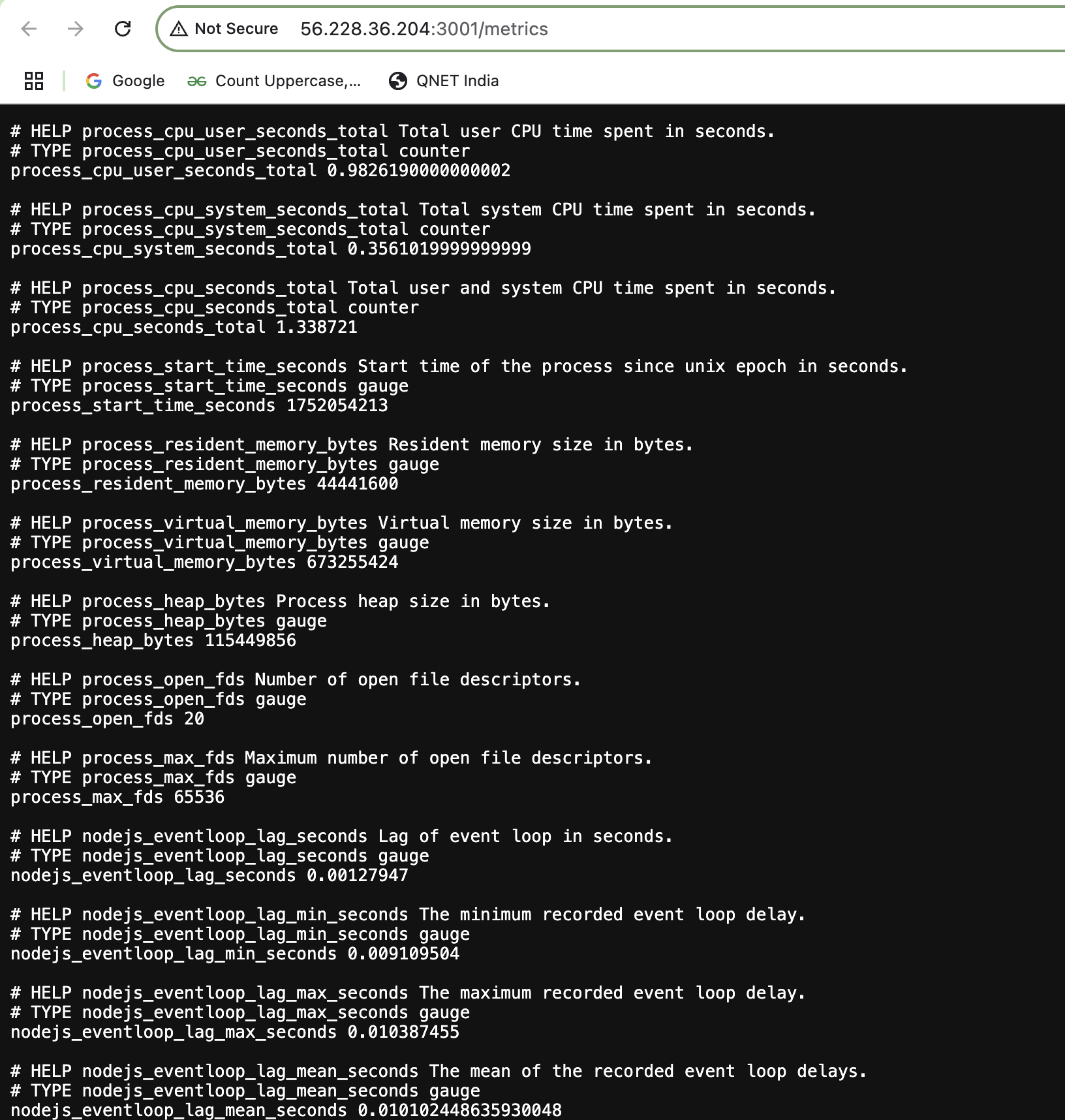


**Prometheus Metrics:**

**Metrics endpoint in Node.js:** Prometheus is used to **collect real-time performance metrics** from your running Node.js application.  
These metrics are essential for **monitoring**, **alerting**, and **visualizing health trends** in Grafana.

**Implementation in Node.js:** You integrate Prometheus in your Express app using the **prom-client** package.

Prometheus Metrics at: http://56.228.36.204:3001/metrics



**Grafana Monitoring:**

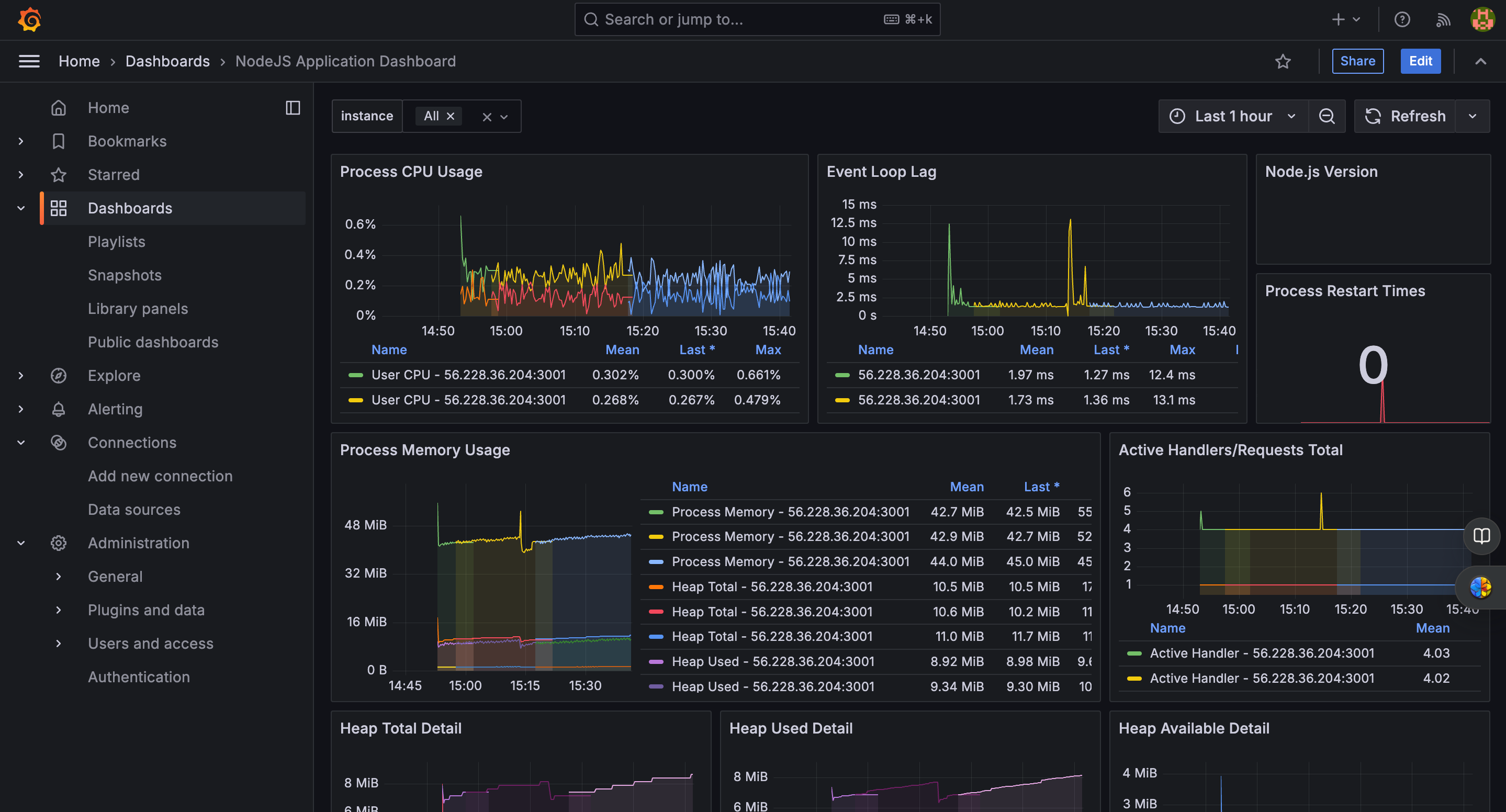
Imported Node.js dashboard from ID 11159**,** Monitors CPU, memory, event loop, heap, and requests.

Grafana is a powerful open-source analytics & visualization tool used to **monitor real-time metrics** collected by Prometheus. It provides **interactive dashboards** to visually analyze system performance, resource usage, and service health.

**Smart Alerts (Prometheus + Alertmanager):** Smart alerts help **automatically detect and notify** you when something goes wrong in your application or infrastructure. This includes things like:

* High CPU or memory usage
* Application downtime
* Slow response times
* Unexpected traffic patterns

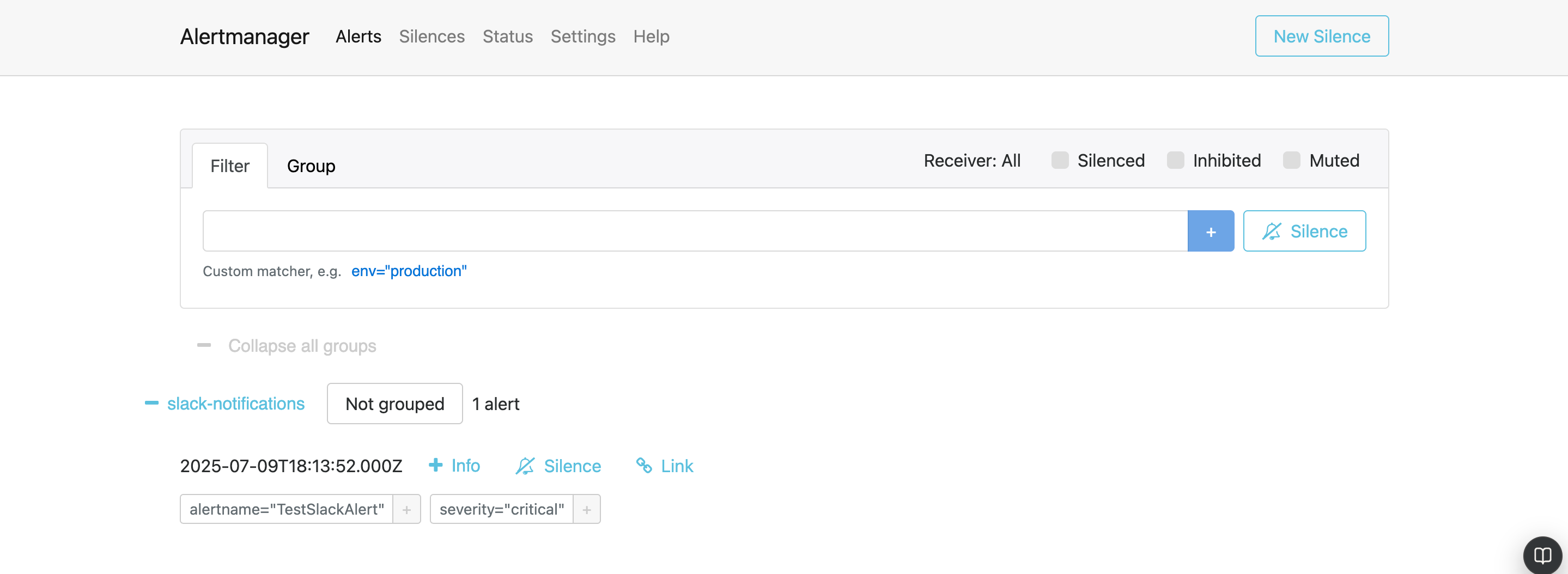
These alerts enable proactive monitoring and quick action before end users are impacted.

Grafana Dashboard:  


**Innovation- AI-Based Anomaly Detection (Grafana ML Plugin):**

While traditional monitoring uses **fixed threshold alerts**, AI-based anomaly detection **automatically learns** from historical data to identify:

* Sudden spikes or drops in CPU, memory, traffic, or error rates
* Out-of-pattern behaviors (e.g., a metric behaving abnormally even within a valid range)
* Issues even before fixed thresholds are breached

This reduces **false positives** and catches **early indicators** of failure.  


**Deployed At:**

* EC2 Staging IP: http://56.228.36.204:3001
* Prometheus: http://localhost:9090
* Grafana: http://localhost:3000
* Jenkins: <http://localhost:8080>

**Future Enhancements:**

**1. Slack/Email Integration with Alertmanager: Current**: Basic Prometheus alerting setup.

**Enhancement**:Configure **Slack channels** and **Email alerts** using Alertmanager’s receiver configurations.Send high-priority alerts for security threats, failed builds, or resource spikes.Ensure **instant team notification** for faster incident response.

**2. Auto-Triggered Jenkins Jobs on Anomaly: Current**: Jenkins pipelines are triggered by code changes only.

**Enhancement**: **Connect Grafana ML plugin** or Prometheus alerts to trigger Jenkins jobs dynamically.  
For example: Restart service on high memory usage.  
Trigger security scan if suspicious traffic spikes.

Use **Jenkins Webhooks/API calls** for execution.

**3. Database Monitoring**: **Current**: No direct monitoring of database performance.

**Enhancement**:Export DB metrics using tools like:

* mysqld\_exporter for MySQL/MariaDB
* postgres\_exporter for PostgreSQL

Visualize:

* Query execution time  
  Connection count  
  Cache hit ratio  
  Set alerts for **slow queries**, **high replication lag**, or **deadlocks**.

**4. Dynamic Auto-Scaling**

* Configure EC2 Auto Scaling or integrate with Kubernetes (EKS) for scaling containers based on:
  + CPU usage
  + Memory
  + Custom metrics like /metrics response time

**5. Smart Prediction with AI**

* Integrate **Grafana’s ML plugin** to:
  + Predict unusual trends in traffic, CPU, or memory
  + Preemptively scale or alert teams
  + Detect **zero-day behavior** anomalies