NISHA N C

Full CI/CD Setup for .NET App on Azure AKS

The project is split into 3 phases. here is what we are doing

Phase 1

INFRASTRUCTURE SET UP

- AKS Cluster
- SonarQube on AKS
- Jenkins via a VM
- Hashicorp Vault on AKS
- Trivy and Git Leaks
- Monitoring (Prometheus and Grafana) on AKS

Phase 2

Setting up repository

- Capstone CI repo
- Capstone CD repo
- IAC (Terraform files for AKS cluster)

Phase 3

CI/CD Pipeline

- Automated with webhook trigger
- Security integration with every step of the pipeline

Tech Tools:

1. .NET
2. MongoDB
3. Terraform
4. Jenkins
5. SonarQube
6. trivy
7. Gitleaks
8. Vault
9. Prometheus
10. Grafana
11. Kubernetes (AKS)

INFRASTRUCTURE SET UP

01

Note: We are setting this up on a separate VM naming it Installer VM

AKS Setup

1. SSH into your installer VM. update and upgrade
2. Install Azure CLI and configure it.
3. Install Terraform
4. Git clone your repository with tf files for AKS
5. Install Kubectl
6. initialize your AKS tf files and apply
7. verify using Azure CLI .
8. Install Helm
9. Installs the Ingress-NGINX controller. Acts as a reverse proxy . Supports routing
10. Open the necessary ports

Setting up SonarQube on Azure AKS using Helm

NISHA N C

Title: What is SonarQube?

- SonarQube is an open-source platform for continuous inspection of code quality.
- Supports static code analysis to detect bugs, code smells, and security vulnerabilities.
- Commonly integrated into CI/CD pipelines.

To set up on your aks cluster:

1. A separate directory and namespace for SonarQube (sonarqube)
2. A StorageClass available (managed-csi) on azure during your aks set up
3. Adds SonarQube's official Helm chart repository
   - Updates local Helm repo cache
4. Custom values.yaml – Service Setup

- Creates an Azure LoadBalancer service or node port to expose SonarQube
- Allows access via ports 9000 (UI) and 9001 (HTTPS)
- Deploys a managed PostgreSQL DB alongside SonarQube
- Persistent volume for PostgreSQL data

MongoDB StatefulSet Deployment in Kubernetes

Goal: Deploy a highly available MongoDB Replica Set in a Kubernetes cluster using StatefulSet.

Headless Service (mongo)

Enables stable DNS (mongo-0.mongo.webapps.svc.cluster.local) for replica set members.

StatefulSet (mongo)

- Replicas: 3 — Creates mongo-0, mongo-1, mongo-2
- Uses persistent storage (managed-csi volume) per pod
- MongoDB runs with --replSet rs0 to support replication
- Probes check MongoDB health (ping via mongosh)

Init Container (on mongo-0 only)

- Temporarily starts MongoDB
- Initializes the Replica Set (rs.initiate(...))
- Adds mongo-1 and mongo-2 as members
- Shuts down after init

To Verify

1. Run MongoDB client in Kubernetes:
2. Check Replica Set Status:

HashiCorp Vault on AKS (Production-Ready Setup)

NISHA N C

1. Create Namespace for Vault
    - Isolate Vault into its own namespace for RBAC & resource separation
2. Add Helm Repo & Install Vault with Raft HA
    - Use Helm and custom values for highly available Raft storage
3. Configure Raft Storage in vault-values.yaml
    - Enables internal HA consensus with retry join logic
4. Disable Pod Anti-Affinity (Optional in Production)
    - By default, Vault pods spread across different nodes.
    - To allow multiple pods on the same node, disable anti-affinity
5. Enable Vault UI and Injector
    - ui.enabled = true for web interface
    - injector.enabled = true to inject secrets into workloads automatically
6. Initialize and Unseal Vault
    - Run vault operator init once
    - Use any 3 of 5 unseal keys per pod to activate
7. Configure Auth Backend & Store Secrets
    - Get Kubernetes API data & configure Vault
    - Enable KV v2 engine and store secrets like MongoDB connection string

Monitoring Setup in Azure AKS (Using Prometheus + Grafana via Helm)

1. Add Prometheus Helm Chart Repo
2. Define Custom values.yaml
    - Prometheus with managed-csi PVC
    - Grafana enabled with nodeport
    - Node Exporter, Kube State Metrics for system & Kubernetes metrics
    - Basic auth: admin / admin123
    - Extra scrape jobs configured
3. Deploy Monitoring Stack
4. Patch Services for External Access as node port (due to our public ip availability constraints)
5. Access Dashboards & Monitor EKS
    - 🌐 Grafana → View dashboards
    - 📡 Prometheus → Scrape metrics
    - 📈 Real-time cluster monitoring for pods, nodes, and apps

Jenkins Setup On Ubuntu Virtual Machine

Install Java (OpenJDK 21)

Required for Jenkins to run.

Install Jenkins

Add Jenkins repo and GPG key
Install and enable Jenkins service

Install Security Scanning Tools

- Gitleaks – Detects hardcoded secrets
- Trivy – Scans containers, images & IaC

Install .NET 8 SDK

Add Microsoft repo, dependencies
Install .NET 8 for building & running C# apps

Install Docker & Configure Jenkins Access

This space is for a quick description of each of the things on the slide. Help people understand what each is about.

Open Required Ports (network Security Group)

22 – SSH
8080 – Jenkins

Setting Up repositories

02

For our CI CD pipeline

PHASE 2

We have three repositories in our github repo

Capstone CI repo

A repository with .Net program files and docker-compose file. This will be used to build our application in our CI pipeline with a webhook trigger so that whenever there is a change in the commit it gets automatically built

Capstone CD repo

CD repo contains Kubernetes manifests (Deployment, Service, Ingress) that deploy and expose your Dockerized .NET app to a Kubernetes cluster. It also integrates Vault secret injection using annotations and a bound service account for secure runtime configuration.

Terraform files for AKS

Terraform files to deploy a AKS cluster on AKS

CI/CD pipeline

Automating security at every point

03

Capstone CI Pipeline Overview (Jenkins)

1. Code Checkout & Security Scans

- Clone from GitHub main branch
- Run GitLeaks for secret detection
- Perform Trivy filesystem scan

2. Build & Test

- Clone from GitHub main branch
- Run GitLeaks for secret detection
- Perform Trivy filesystem scan

3. Docker Build & Image Scanning

- Build Docker image: nishanc7/noteapp: ${BUILD_NUMBER}
- Run Trivy image scan on the Docker image

4. Push Image & Trigger CD

- Push Docker image to DockerHub
- Auto-update the manifest.yaml in CD repo with new tag
- Commit and push to CD repo: triggers deployment

5. Notifications

- Email pipeline status with color-coded HTML banner
- Includes job name, build number, and result link

Capstone CD Pipeline Overview (Jenkins)

1. Git Checkout

Pulls the latest manifest files from the CD repo: Capstone-Dotnet-manodb-CD.git (main branch)

2.Deploy to Kubernetes (AKS)

- Applies the following manifests to the webapps namespace:
  - manifest.yaml (Deployment + Service)
  - ci.yaml (ConfigMap/Secret/Vault Agent?)
  - ingress.yaml (External access)
- Uses withKubeConfig and Jenkins credential (k8s-token) to connect to your cluster securely.

3. Verify Deployment

- Checks deployment status via:
  - kubectl get pods
  - kubectl get svc
  - kubectl get ingress
- Waits 30 seconds for resources to stabilize

4. Notifications

Sends an HTML email with:

- Pipeline name, build number, status
- Success = green banner, Failure = red banner
- Link to console output

You're building an end-to-end DevOps pipeline for a production-grade .NET web application using modern tools and best practices. Here's what you're proving through this project:

What You're Trying to Achieve

✅ 1. CI/CD Automation

- You're showing that you can:
- Build, scan, test, and containerize code automatically Push images to DockerHub
- Deploy those images to Kubernetes zero manual steps

✅ 2. Secure, Cloud-Native Deployment

You're using:

- Vault to securely inject secrets at runtime (no hardcoded secrets)
- Ingress + Services for external access and internal routing
- Kubernetes as your orchestration platform (AKS)

✅ 3. DevSecOps Practices

You're demonstrating security-first pipelines with:

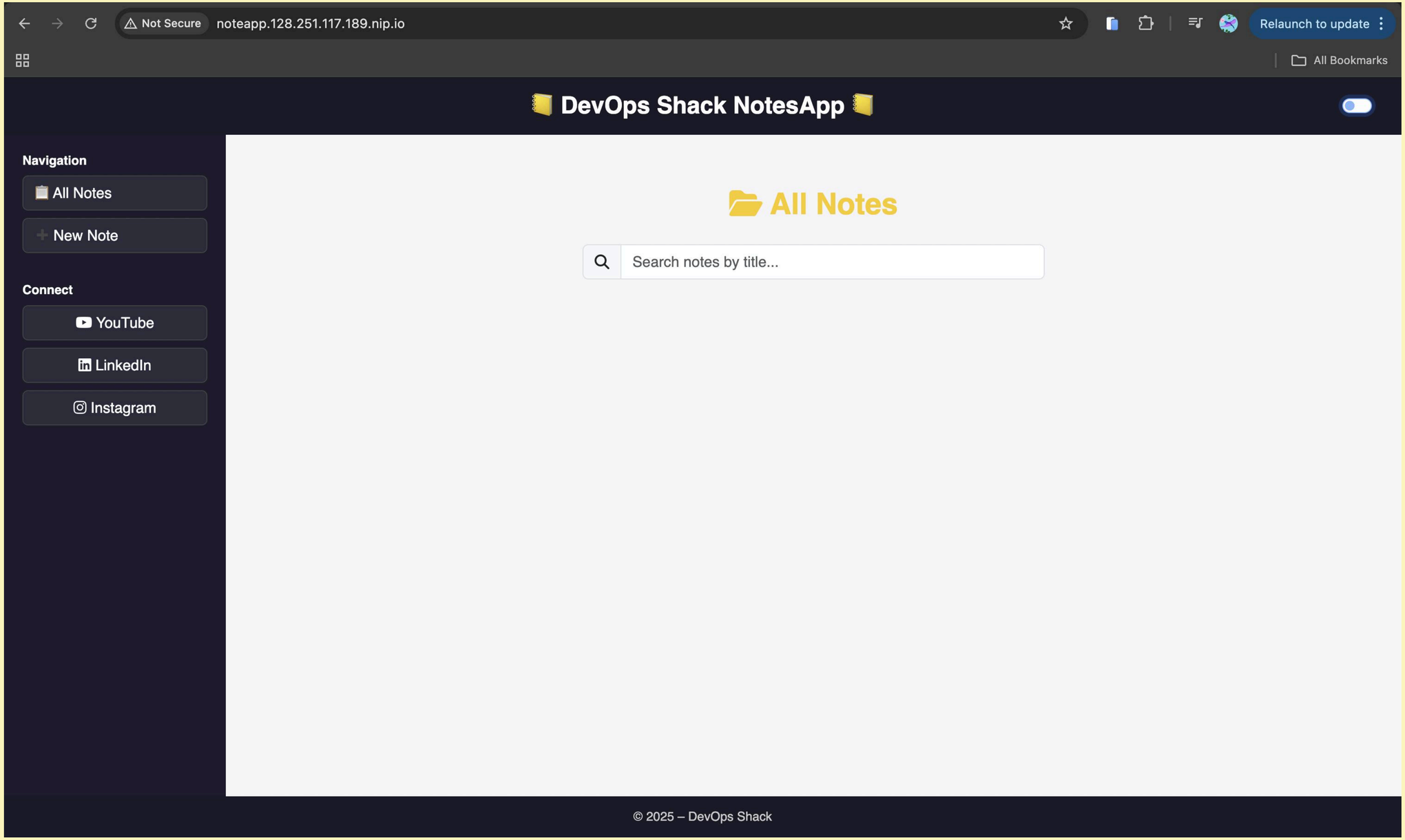
- GitLeaks (secrets detection)
- Trivy (vulnerability scans on code & images)
- SonarQube (code quality gate enforcement)

✅ 4. Real-World Tools & Skills

This project proves you're comfortable with:

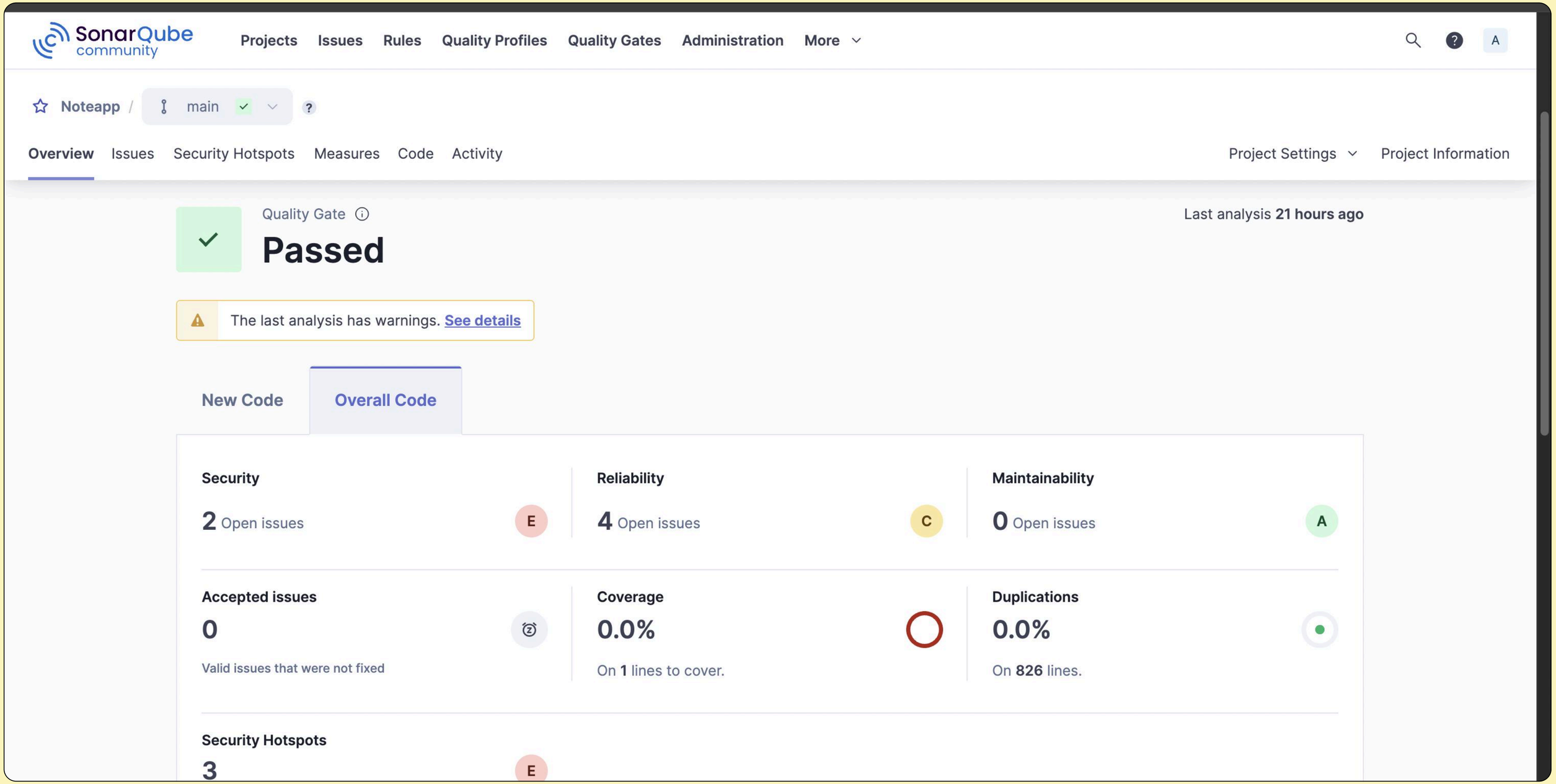
- Jenkins
- Docker & Kubernetes
- Helm, GitHub, Bash, YAML
- Cloud platforms (Azure)

Uh Oh! Problem	Why is this happening	A Workaround Used...
Networking & Public IP Constraints in Azure AKS only had 3 public IPs available in the region	By default, each LoadBalancer service consumes one public IP. Azure enforces a soft quota for public IPs per region unless requested to increase.	Used NodePort services to expose applications. Did kubectl port-forward from a VM that had access to the cluster.
I Can’t buy a domain to assign it to the AKS Ingress controller to access it via internet.	As a result, DNS-based domain configuration was not practical in the current setup.	• Used .nio direct DNS resolution: • A local DNS routing solution for testing and previewing the deployed app. • Enabled access without assigning a domain name.
Repeated Forbidden errors while applying Kubernetes manifests	The noteapp-sa ServiceAccount lacked permissions to read and manage cluster resources.	Created Role & RoleBinding ClusterRoleBinding for accessing cluster-wide resources (e.g., PersistentVolumes, ClusterIssuers). Scoped RoleBindings to webapps namespace for better security and isolation.
Jenkins Pipeline Failures	• Either i made a spelling mistake • I didn’t open certain ports • Wrong credentials • Kubectl not found on my machine • API calls failed due to i/o timeout or wrong protocol	Patiently fix it. Look where you spelled it wrong Use azure cli to set up nsg and inspect Use HTTPS and ensure Jenkins VM is on the same subnet or has proper NSG rules.



Deployed it successfully :)

SONARQUBE ANALYSIS - SUMMARY
TO PERFORM STATIC CODE ANALYSIS ON YOUR .NET BASED NOTEAPP USING SONARQUBE IN A JENKINS CI PIPELINE.



SonarQube + Quality Gate Summary

⚡ Codebase Stats:

- 412 files indexed
- 7 languages detected:
  cs, css, docker, js, json, web, yaml

🔥Analysis Performed:

- HTML analysis
- JavaScript (including inside YAML/HTML)
- CSS Rules & Metrics
- Dockerfile inspection (IaC)
- IaC Kubernetes & Azure ARM scanning
- Text secrets detection
- Git SCM detection (partial)

🧪 Metrics Summary (from log insights)

Metric	Value
Files Scanned	412
Languages Detected	7
Coverage Info	❌ Not available
Code Smells	✅ No critical ones reported
Security Hotspots	✅ None flagged
Duplication (CPD)	✅ Checked
SCM Info	⚠️ Partial (missing for `/obj/`)

📈 Quality Gate Summary

✅ Status: Passed

Project meets the quality standards defined in the "Sonar Way" default profile.

No critical issues, blockers, or maintainability problems found in the scanned code.

GitLeaks Scan Result

used the following command in the CI pipleine

```
gitleaks detect --report-format json --report-path gitleaks-report.json --exit-code 1
```

What it does:

- --report-format json: Saves results in a machine-readable JSON format.
- --report-path gitleaks-report.json: Outputs the scan report to a file.
- --exit-code 1: Fails the pipeline if any secret is found (which is good for security enforcement).

Output from Jenkins:

INF 5 commits scanned.
INF scan completed in 33.8ms
INF no leaks found

✔ Result: No secrets or sensitive data were found in your codebase.

💡 Why It's Important

GitLeaks helps you automate secret detection before they reach production or public repositories. Including it in your pipeline:

- Prevents credential leaks
- Enforces secure development practices
- Reduces risk of data breaches

Trivy FS Scan Summary

used the following command in the CI pipleine

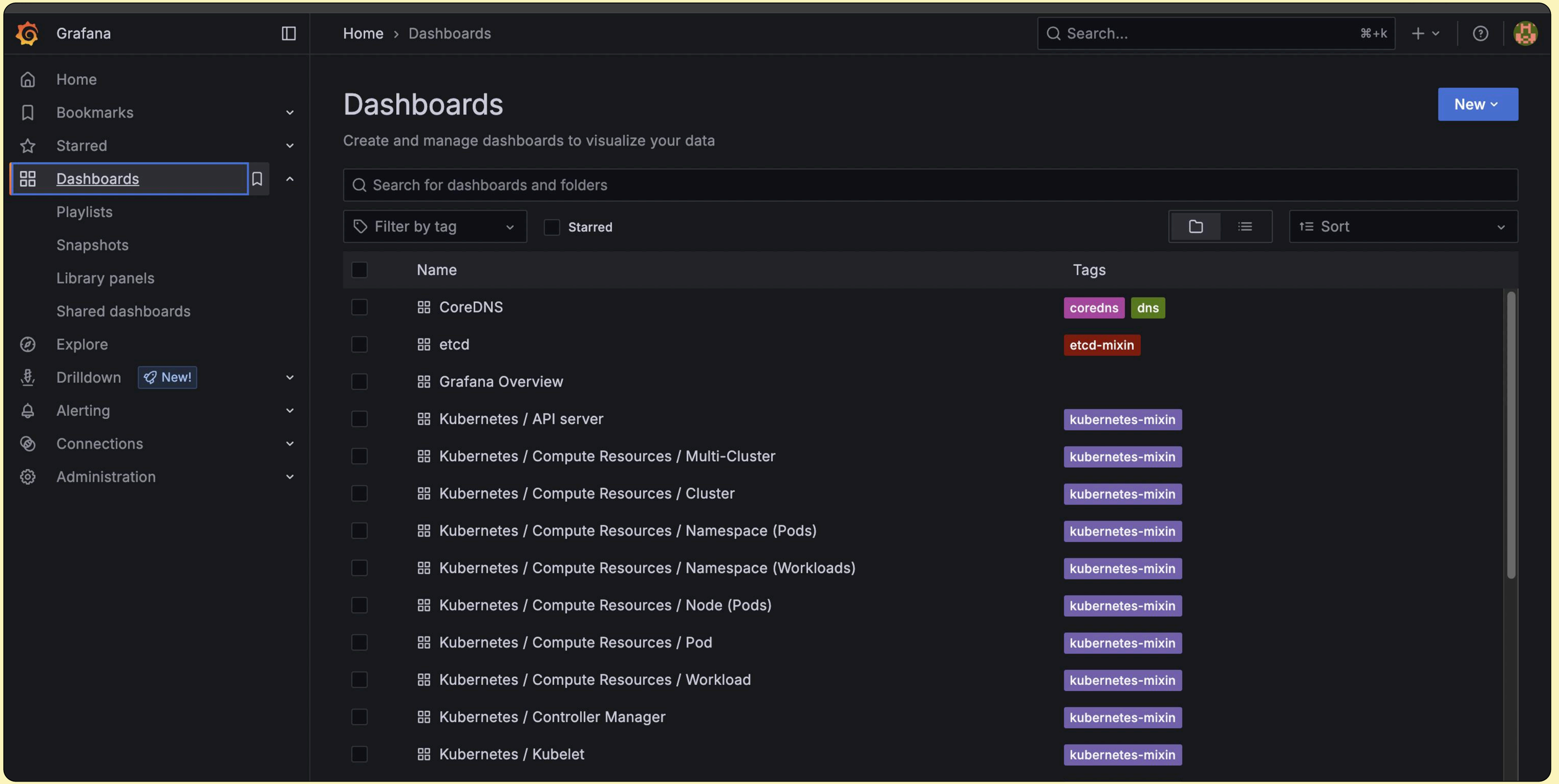
```
trivy fs --format table -o trivy-fs-report.html .
```

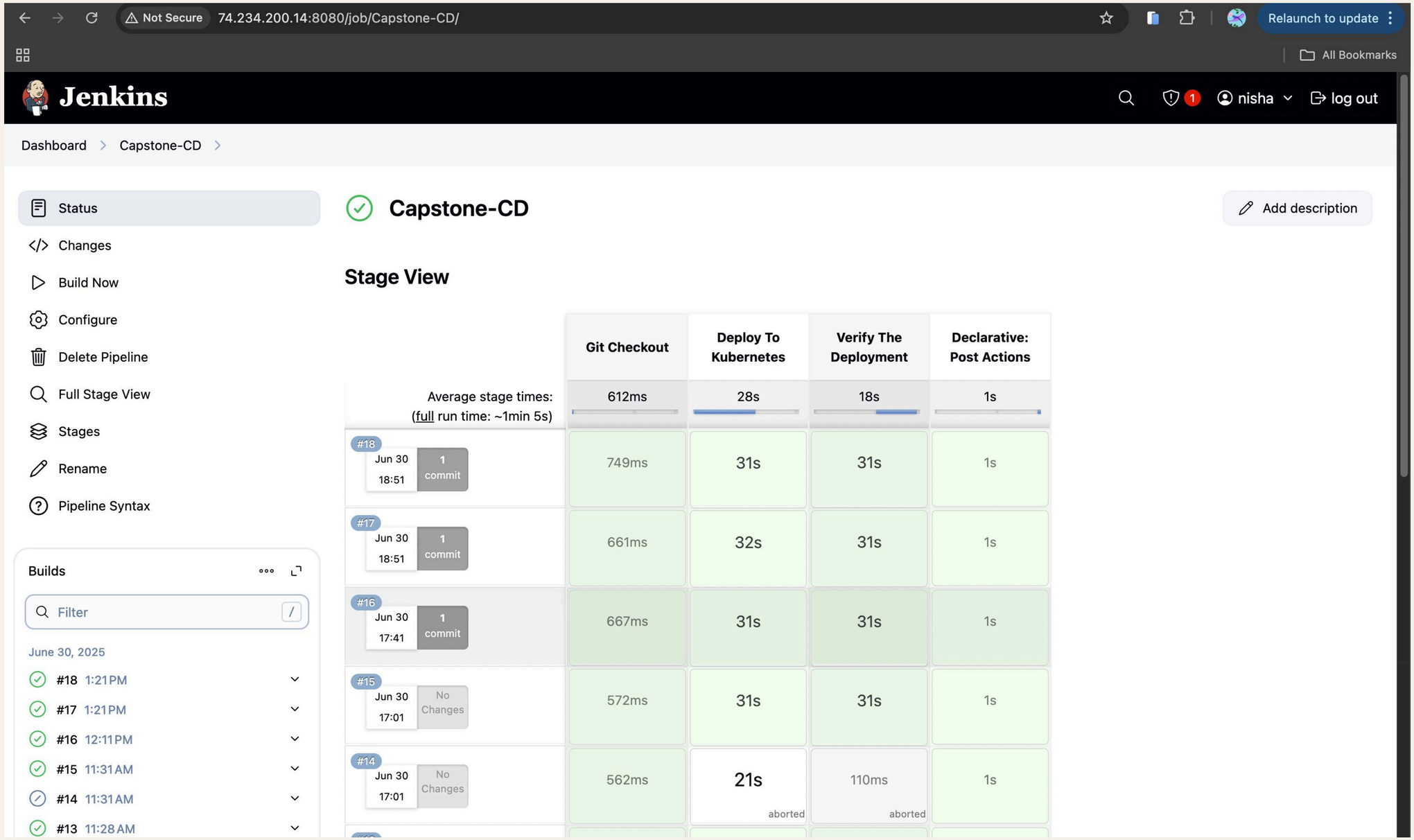
What it does:

- Scan the entire file system in the current directory (.).
- Output the results in table format.
- Save the report as an HTML file (trivy-fs-report.html).

✅ Vulnerability scanning is enabled
→ Trivy is checking for known CVEs in your OS and software dependencies.
🔐 Secret scanning is enabled
→ It is also looking for sensitive data like passwords, API keys, tokens, etc.
⚠️ If your scanning is slow...
→ Tip: You can disable secret scanning (--scanners vuln) to make it faster if needed.
📦 Number of language-specific files: 1
→ Likely a .csproj or similar file related to your .NET project.
🔍 Detecting vulnerabilities with dotnet-core scanner
→ Scanning your .NET project files and dependencies for security issues.

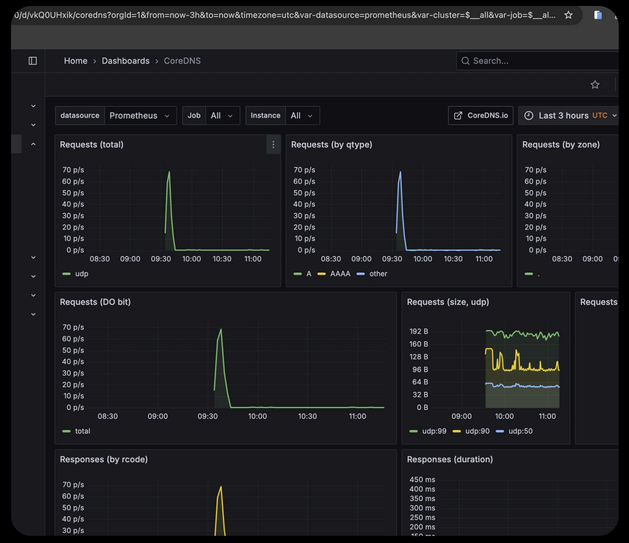
USE AN SSH TUNNEL TO YOUR LOCAL MACHINE
IN A NEW TERMINAL ON YOUR LOCAL MACHINE, FORWARD THE REMOTE VM'S PORT 3000 TO YOUR OWN:



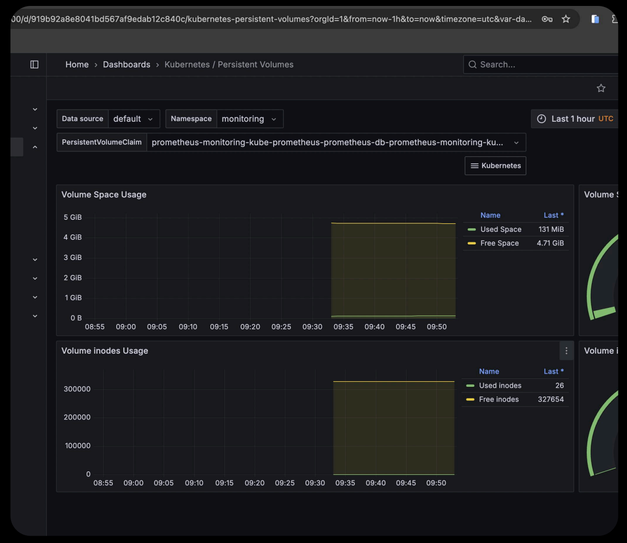


Jenkins CD pipeline

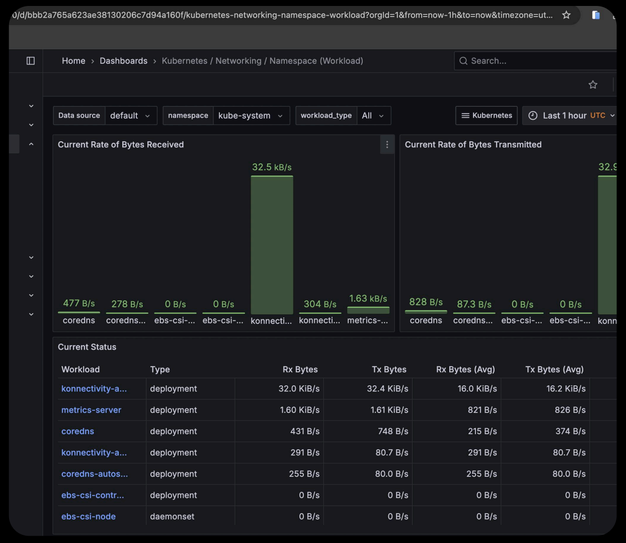
Dashboards created:



A. Networking/ Namespace



B. Persistent Volume



C. Core DNS

Thank you!