

PHASE 4-TESTING

PROJECT TITLE :- Setting up a CI/CD pipeline for automated deployment

COLLEGE NAME:-Dr. SMCE

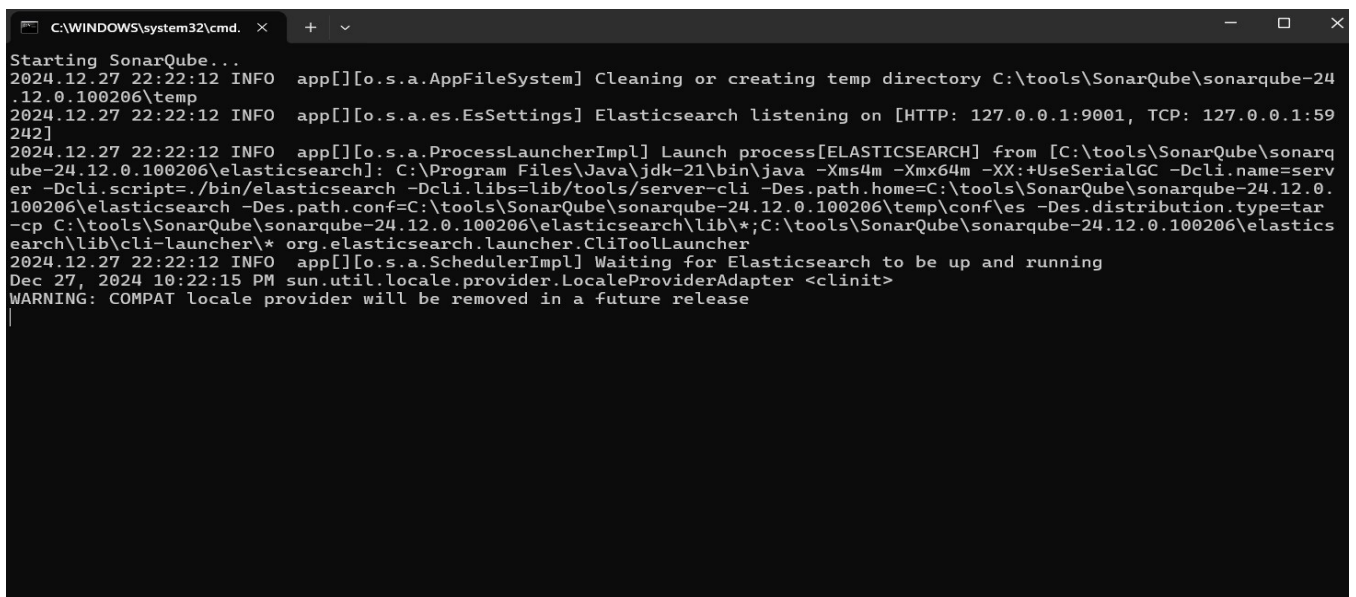
GROUP MEMBERS:-

1. Amith C Y [USN:- 1CC21CS004]
 2. Charan K M [USN:- 1CC21CS017]
 3. Nitin Teja N[USN:-1CC21CS040]
 4. Sunil C M [USN:-1CC21CS057]
-

Phase 4: TESTING

Step 1: Static Code Analysis (SAST)

- Run **SonarQube** to detect security vulnerabilities in the source code.
- Fix issues before progressing to the build stage.



```
C:\WINDOWS\system32\cmd. x + v
Starting SonarQube...
2024.12.27 22:22:12 INFO app[][o.s.a.AppFileSystem] Cleaning or creating temp directory C:\tools\SonarQube\sonarqube-24.12.0.100206\temp
2024.12.27 22:22:12 INFO app[][o.s.a.es.EsSettings] Elasticsearch listening on [HTTP: 127.0.0.1:9001, TCP: 127.0.0.1:59242]
2024.12.27 22:22:12 INFO app[][o.s.a.ProcessLauncherImpl] Launch process[ELASTICSEARCH] from [C:\tools\SonarQube\sonarqube-24.12.0.100206\elasticsearch]: C:\Program Files\Java\jdk-21\bin\java -Xms4m -Xmx64m -XX:+UseSerialGC -Dcli.name=server -Dcli.script=./bin/elasticsearch -Dcli.libs=lib/tools/server-cli -Des.path.home=C:\tools\SonarQube\sonarqube-24.12.0.100206\elasticsearch -Des.path.conf=C:\tools\SonarQube\sonarqube-24.12.0.100206\temp\conf\es -Des.distribution.type=tar -cp C:\tools\SonarQube\sonarqube-24.12.0.100206\elasticsearch\lib\*;C:\tools\SonarQube\sonarqube-24.12.0.100206\elasticsearch\lib\cli-launcher\* org.elasticsearch.launcher.CliToolLauncher
2024.12.27 22:22:12 INFO app[][o.s.a.SchedulerImpl] Waiting for Elasticsearch to be up and running
Dec 27, 2024 10:22:15 PM sun.util.locale.provider.LocaleProviderAdapter <clinit>
WARNING: COMPAT locale provider will be removed in a future release
```

SAMPLE JENKINS FILE

```
pipeline {
    agent any
    environment {
        IMAGE_NAME = "my-python-app"
        DOCKER_HUB_REPO = "docker.io/myusername/my-python-app"
    }
}
```

```

stages {
  stage('Checkout') {
    steps {
      git 'https://github.com/myusername/my-python-app.git'
    }
  }

  stage('Code Quality Check') {
    steps {
      withSonarQubeEnv('SonarQube') {
        sh 'sonar-scanner'
      }
    }
  }

  stage('Build Docker Image') {
    steps {
      sh 'docker build -t $IMAGE_NAME .'
    }
  }

  stage('Push to DockerHub') {
    steps {
      withCredentials([usernamePassword(credentialsId: 'dockerhub-credentials',
usernameVariable: 'DOCKER_USER', passwordVariable: 'DOCKER_PASS')]) {
        sh """
          echo $DOCKER_PASS | docker login -u $DOCKER_USER --password-stdin
          docker tag $IMAGE_NAME $DOCKER_HUB_REPO
          docker push $DOCKER_HUB_REPO
        """
      }
    }
  }

  stage('Deploy to Kubernetes') {
    steps {
      sh 'kubectl apply -f k8s/deployment.yaml'
    }
  }
}

post {
  always {
    mail to: 'devteam@example.com',
        subject: "Build ${currentBuild.fullDisplayName}",
        body: "Pipeline ${currentBuild.fullDisplayName} finished with status:
${currentBuild.currentResult}"
  }
}
}

```

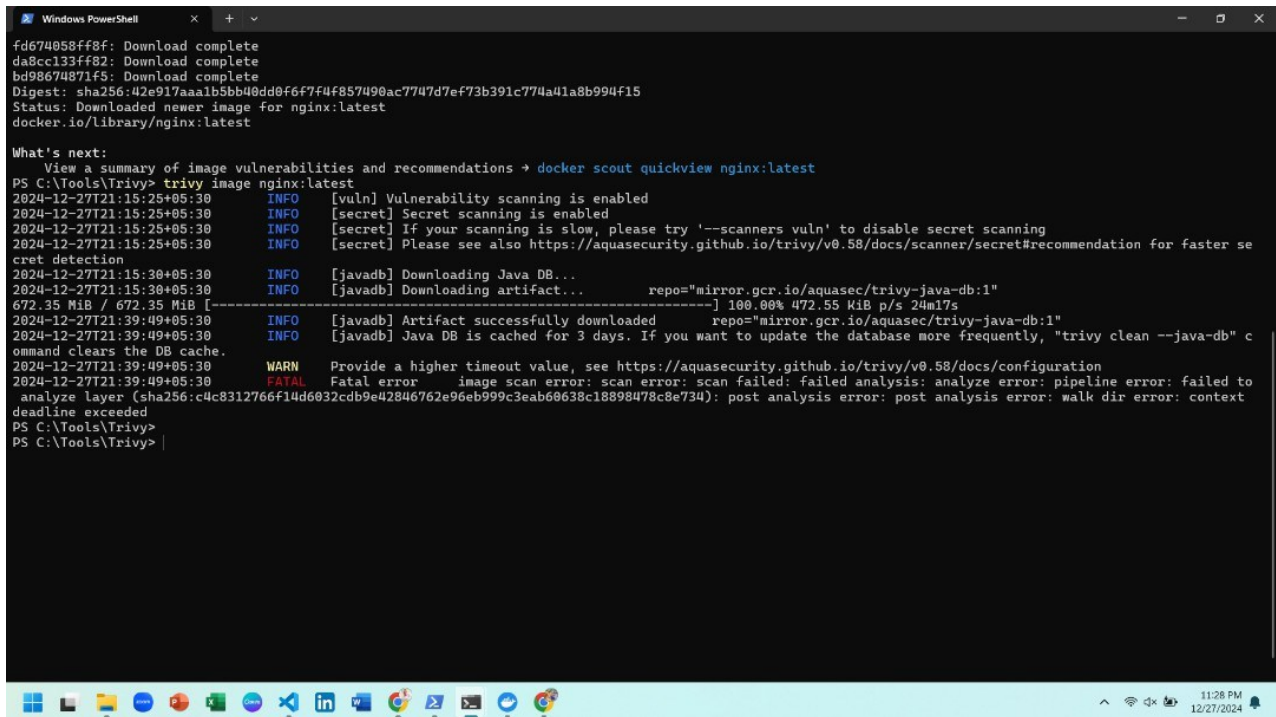
Testing and Validation

The pipeline was rigorously tested across several dimensions:

- **Static Code Analysis:** Validated the integration of SonarQube and verified the accuracy of reported metrics.
- **Build Verification:** Ensured Docker images were successfully built and tagged with the correct version identifiers.
- **Functional Testing:** Unit tests were executed automatically as part of the pipeline using pytest.
- **Deployment Validation:** The containerized Python app was tested post-deployment in a Minikube cluster.
- **Rollback Strategy:** Kubernetes deployment configuration supported rolling updates with rollback on failure.

Step 3: Container Security Testing

- Scan container images using **Trivy** or **Snyk**.
- Enforce least privilege access policies.



```
Windows PowerShell
fd674058ff8f: Download complete
da8cc133ff82: Download complete
bd98674871f5: Download complete
Digest: sha256:42e917aa1b5bb40dd8f6f7f4f857490ac7747d7ef73b391c774a41a8b994f15
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest

What's next:
View a summary of image vulnerabilities and recommendations → docker scout quickview nginx:latest
PS C:\Tools\Trivy> trivy image nginx:latest
2024-12-27T21:15:25+05:30 INFO [vuln] Vulnerability scanning is enabled
2024-12-27T21:15:25+05:30 INFO [secret] Secret scanning is enabled
2024-12-27T21:15:25+05:30 INFO [secret] If your scanning is slow, please try '--scanners vuln' to disable secret scanning
2024-12-27T21:15:25+05:30 INFO [secret] Please see also https://aquasecurity.github.io/trivy/v0.58/docs/scanner/secret#recommendation for faster se
cret detection
2024-12-27T21:15:30+05:30 INFO [javadb] Downloading Java DB...
2024-12-27T21:15:30+05:30 INFO [javadb] Downloading artifact... repo="mirror.gcr.io/aquasec/trivy-java-db:1"
672.35 MiB / 672.35 MiB [-----] 100.00% 472.55 KiB p/s 24m17s
2024-12-27T21:39:49+05:30 INFO [javadb] Artifact successfully downloaded repo="mirror.gcr.io/aquasec/trivy-java-db:1"
2024-12-27T21:39:49+05:30 INFO [javadb] Java DB is cached for 3 days. If you want to update the database more frequently, "trivy clean --java-db" c
ommand clears the DB cache.
2024-12-27T21:39:49+05:30 WARN Provide a higher timeout value, see https://aquasecurity.github.io/trivy/v0.58/docs/configuration
2024-12-27T21:39:49+05:30 FATAL Fatal error image scan error: scan failed: failed analysis: analyze error: pipeline error: failed to
analyze layer (sha256:c4c8312766f14d6032c0b9e42846762e96eb999c3eab60638c18898478c8e734): post analysis error: post analysis error: walk dir error: context
deadline exceeded
PS C:\Tools\Trivy>
PS C:\Tools\Trivy> |
```

Step 4: Infrastructure and Compliance Testing

- Scan Kubernetes configurations with **Checkov** to detect misconfigurations.
- Verify compliance with **GDPR**, **PCI-DSS**, **SOC2** security policies.

Step 5: CI/CD Pipeline Security Validation

- Ensure security policies are enforced in the **Jenkins pipeline**.
- Monitor deployments for unauthorized changes using **ELK Stack**.

FUTURE IMPROVEMENTS

1. **Enhance Security Automation:** Implement real-time security monitoring and automated patch management.
2. **Adopt Zero Trust Architecture:** Introduce strict authentication and continuous security verification.
3. **Implement Advanced Threat Detection:** Use AI-powered threat detection tools for anomaly monitoring.
4. **Optimize Multi-Cloud Security:** Extend security controls across **AWS, Azure, and Google Cloud.**
5. **Automate Security Audits:** Set up automated penetration testing and compliance validation.