Managing Security Across Multiple Environments with DevSecOps

PHASE 3- SOLUTION DEVELOPMENT AND TESTING

College Name: Shetty Institute of Technology Kalaburagi.

Group Members:

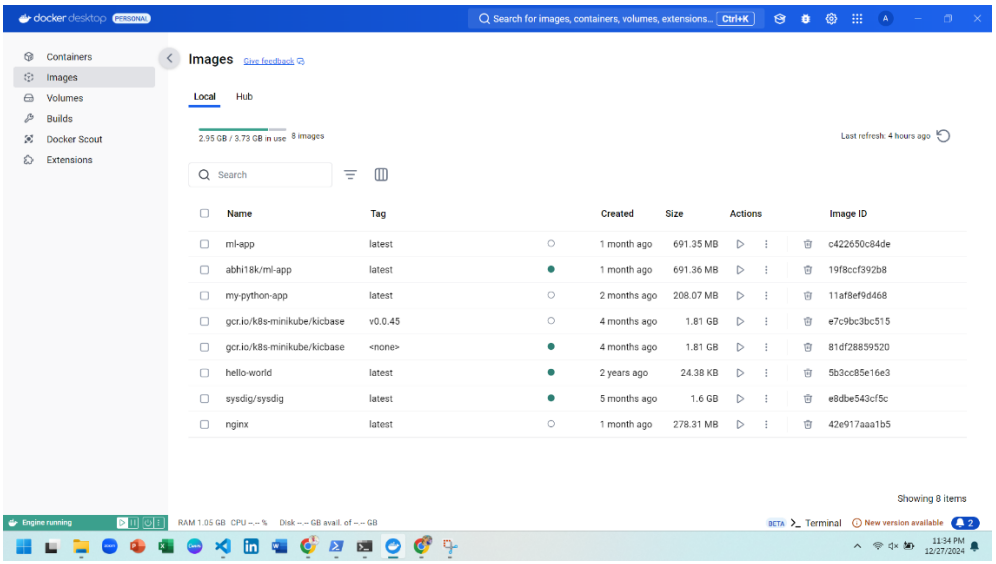
- **Name: ABHISHEK S M**
  CAN ID Number: CAN_33981242
  WORK: SOLUTION DEVELOPMENT
- **Name: BHAGYASHREE M HANDI**
  CAN ID Number: CAN_33254590
- **Name: SHARANU**
  CAN ID Number: CAN_33489513
- **Name: SHASHIKIRAN**
  CAN ID Number: CAN_33287479

SOLUTION DEVELOPMENT:

Implementing Containerization and Running Locally

Step 1: Set Up the Development Environment

1. Install Docker for containerization.



2. Install Kubernetes (Minikube) for local container orchestration.

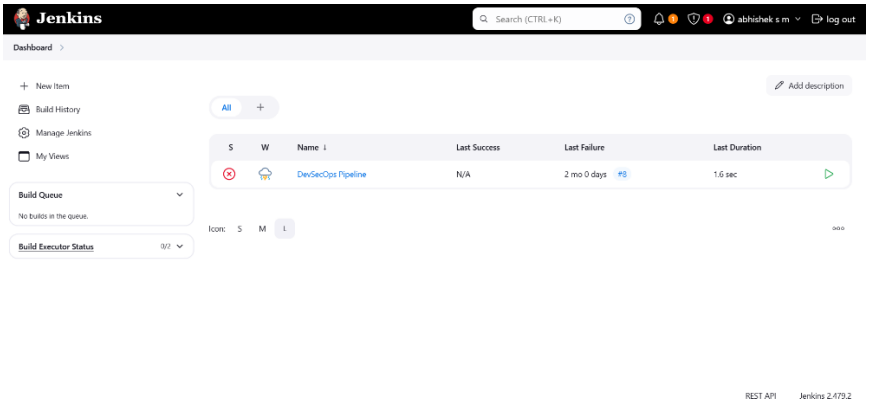
```
C:\Users\abhi\Desktop\Devops\ml-app>minikube start
* minikube v1.34.0 on Microsoft Windows 11 Home Single Language 10.0.26100.2314 Build 26100.2314
* Using the docker driver based on existing profile
* Starting "minikube" primary control-plane node in "minikube" cluster
* Pulling base image v0.0.45 ...
* Updating the running docker "minikube" container ...
! Failing to connect to https://registry.k8s.io/ from both inside the minikube container and host machine
* To pull new external images, you may need to configure a proxy: https://minikube.sigs.k8s.io/docs/reference/n
* Preparing Kubernetes v1.31.0 on Docker 27.2.0 ...
* Verifying Kubernetes components...
  - Using image docker.io/kubernetesui/metrics-scraper:v1.0.8
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
  - Using image docker.io/kubernetesui/dashboard:v2.7.0
* Some dashboard features require the metrics-server addon. To enable all features please run:

        minikube addons enable metrics-server

* Enabled addons: default-storageclass, storage-provisioner, dashboard
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

```
C:\Users\abhi\Desktop\Devops\ml-app>minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
```

3. Install Jenkins for CI/CD pipeline automation.



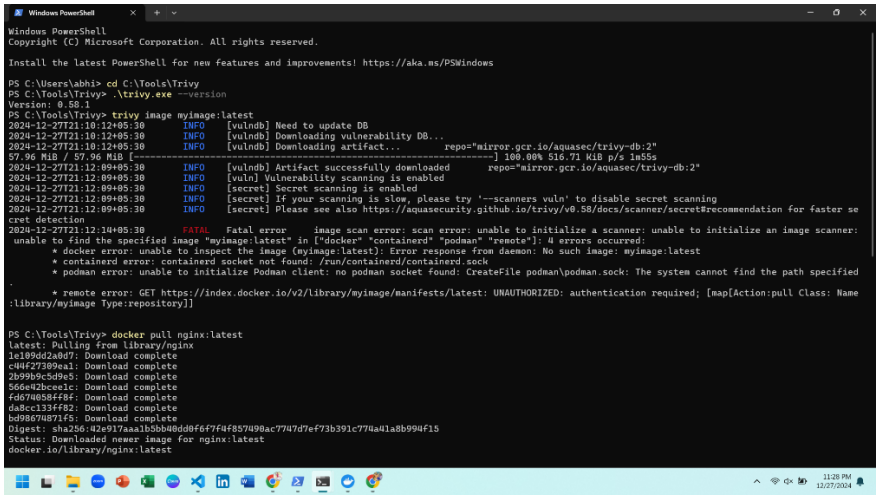
4. Install Git for version control and repository management.

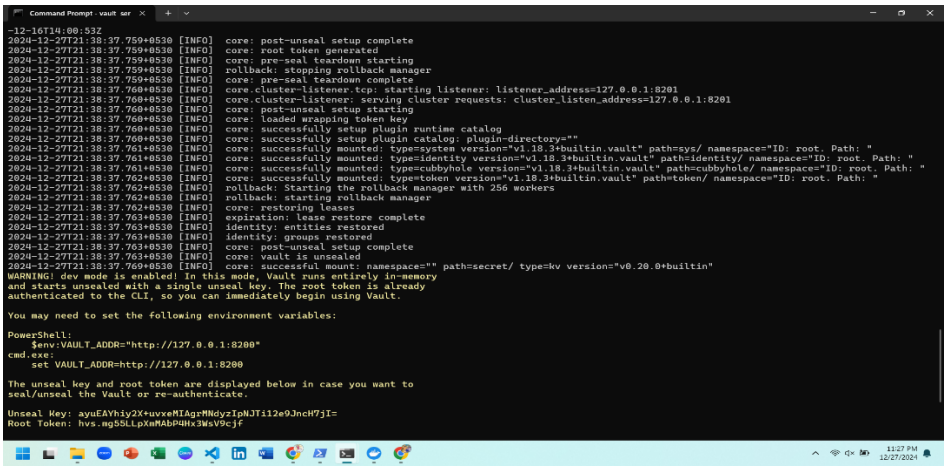
```
Microsoft Windows [Version 10.0.26100.3194]
(c) Microsoft Corporation. All rights reserved.

C:\Users\abhi>git --version
git version 2.47.1.windows.1

C:\Users\abhi>
```

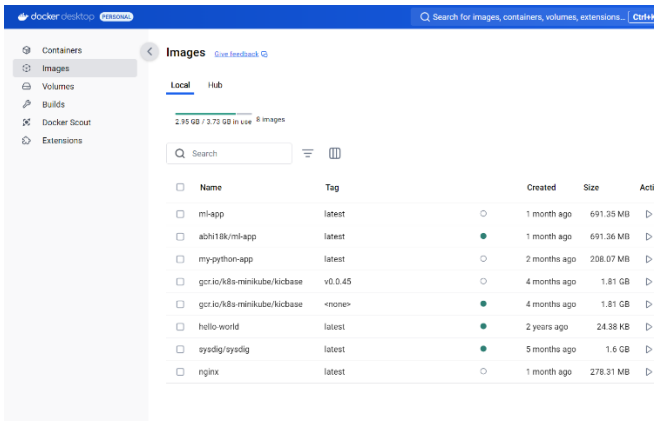
5. Install SonarQube, Trivy, OWASP ZAP for security scanning and vulnerability



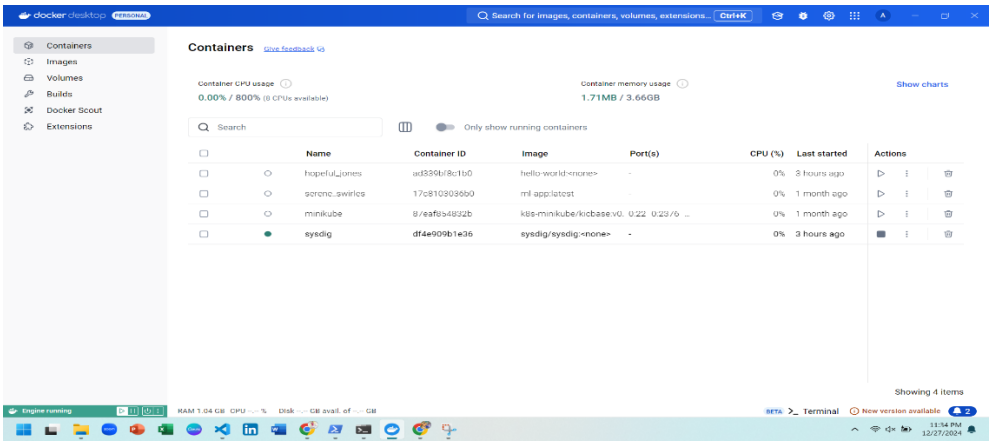


Step 2: Containerizing the Application

1. Create a Docker file for the application:
    - Define the base image (e.g., node:16-alpine).
    - Set up the working directory and dependencies.
    - Expose required ports.
    - Define the startup command (CMD).

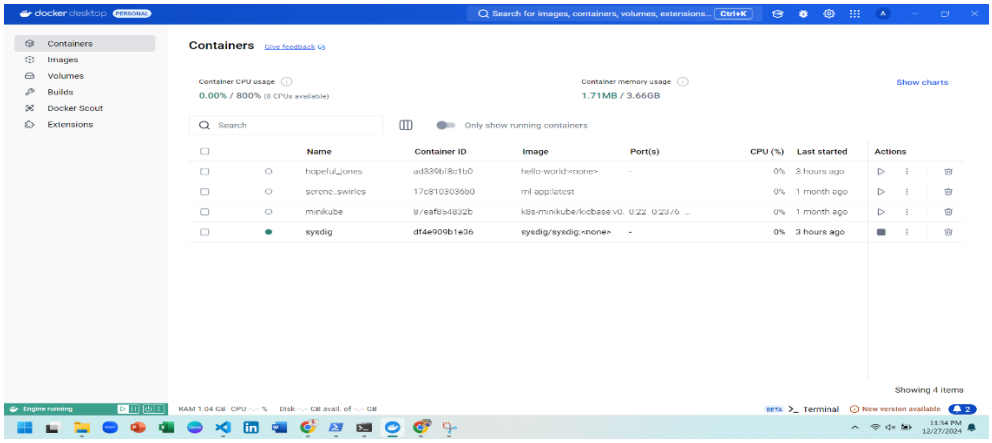


2. Build and run the container locally:



3. docker build -t ml-app .

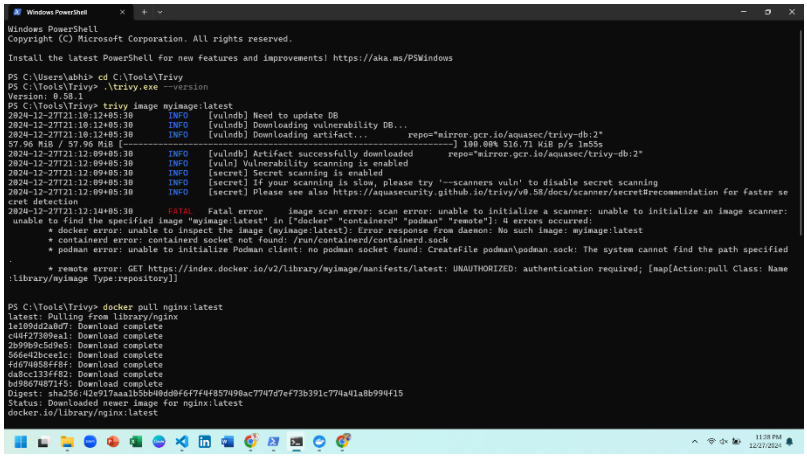
docker run -d -p 8080:8080 ml-app



Step 3: Set Up Container Security

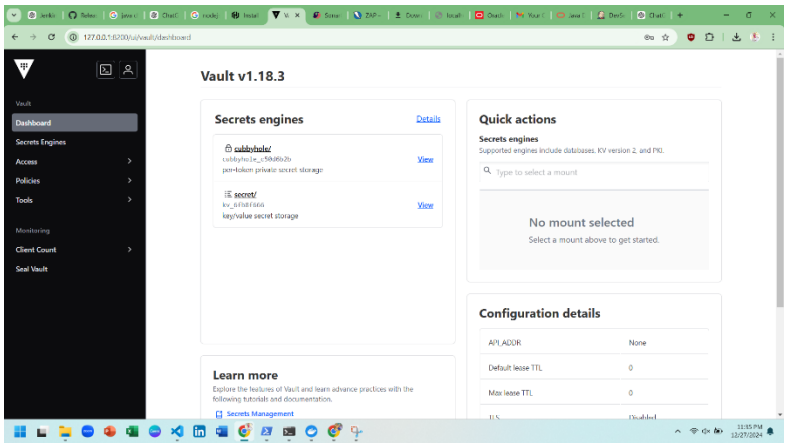
1. Scan container images using Trivy or Snyk.

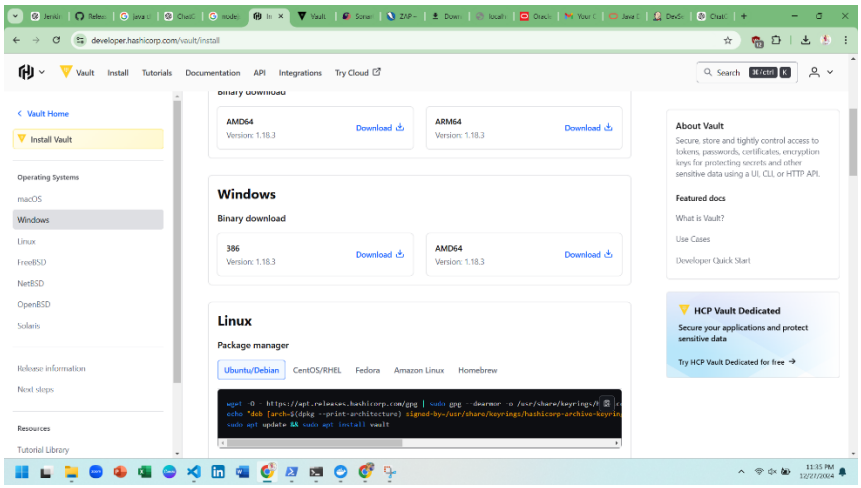
PHASE 3



2. Implement Kubernetes RBAC policies and network security rules.

3. Secure secrets using HashiCorp Vault.





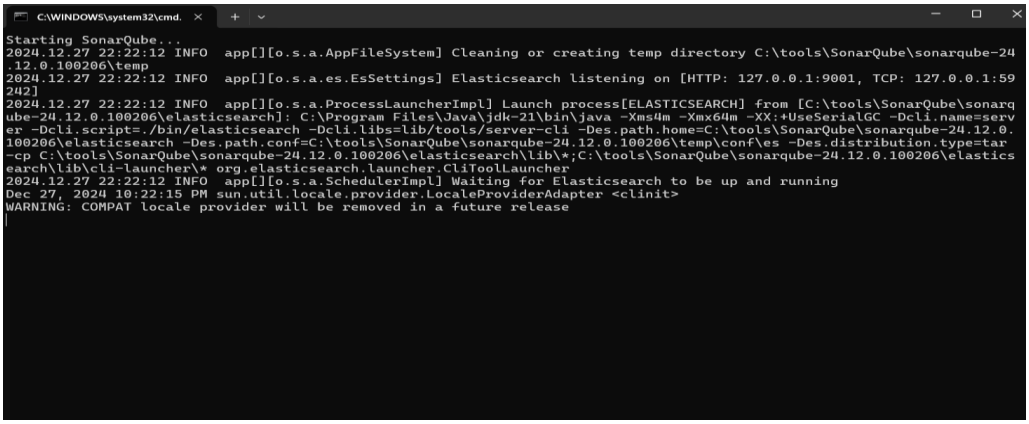
Step 4: Implement CI/CD Security

1. Set up Jenkins for automating CI/CD pipeline.
2. Integrate SonarQube for Static Application Security Testing (SAST).
3. Automate security scanning in the CI/CD pipeline using OWASP ZAP.
4. Deploy to a Kubernetes cluster using Minikube.

SECTION 2: TESTING THE SOLUTION

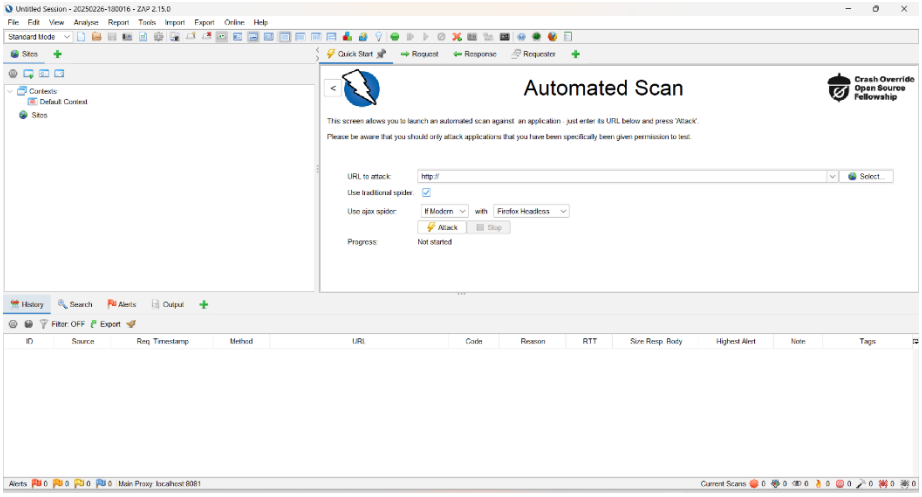
Step 1: Static Code Analysis (SAST)

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



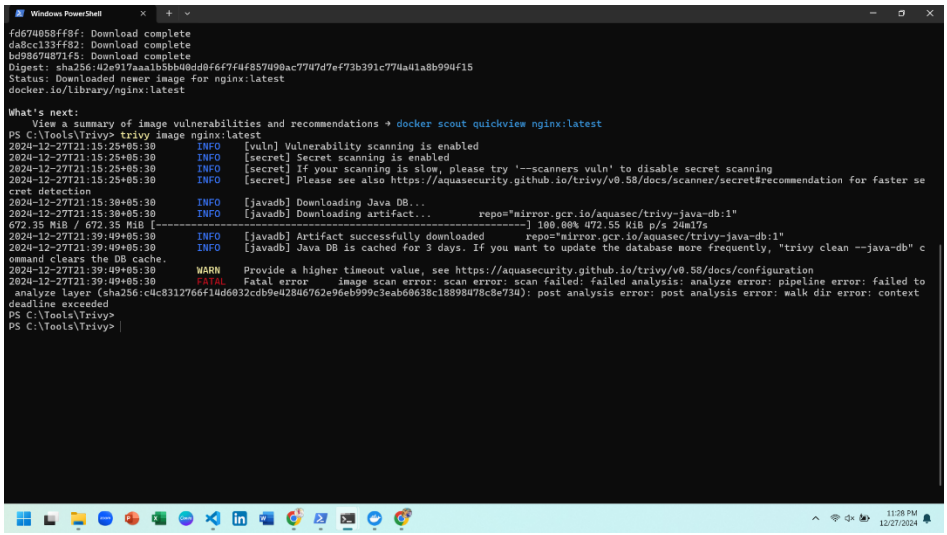
Step 2: Dynamic Application Security Testing (DAST)

- Use **OWASP ZAP** to test for SQL Injection, XSS, and CSRF vulnerabilities.
- Run penetration tests against staging environments.



Step 3: Container Security Testing

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



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.