DevSecOps Project Report

Project Title: Secure CI/CD Pipeline with Monitoring and Testing

---

1. Introduction

Name: *Idah Makena Ncooro*\ **Institution**: *Strathmore University*\ **Project**: *Final DevSecOps Project*

This project represents the final DevSecOps practical assessment, aimed at applying industry best practices in continuous integration, deployment, testing, and monitoring. It showcases a secure, automated CI/CD pipeline for deploying a containerized web application. The implementation integrates tools like Docker, Jenkins, SonarQube, Selenium, and Kubernetes, along with monitoring solutions such as Netdata and AWS CloudWatch. The purpose of this project was not only to deploy and test an application securely but also to gain hands-on experience in managing end-to-end DevSecOps workflows in both local and cloud environments.

---

2. Project Objectives

- Automate builds and deployments using Jenkins
- Scan code quality using SonarQube
- Package and serve the application using Docker and Nginx
- Deploy the application via Helm on Kubernetes
- Monitor the deployment using Netdata and CloudWatch
- Perform UI testing using Selenium and Pytest
- Track development progress using Jira
- Ensure secure deployment using SSL certificates and security headers

---

3. Environment and Tools Setup

Development Tools:

- **VS Code**: Primary code editor
- **PowerShell & Git Bash**: CLI tools
- **GitHub**: Version control
- **Jira**: Agile board and issue tracking

Backend/Infrastructure:

- **Docker Desktop**: Local Kubernetes and Docker support
- **AWS EKS & ECR**: For cloud deployment and container registry • **Jenkins**: CI/CD orchestration
- **SonarQube**: Static analysis (SAST)
- **Helm**: Kubernetes packaging tool
- **Netdata & AWS CloudWatch**: Monitoring solutions
- **Selenium + Pytest**: UI automation testing

---

4. Project Flow

1. **Frontend Development**
2. Created a React project using Vite
3. Styled components using TailwindCSS and shadcn/ui
4. Verified UI using `npm run dev`
5. **Containerization**
6. Created a `Dockerfile` with multi-stage build
7. Final stage uses Nginx to serve the app
8. Configured `.dockerignore`
9. **CI/CD Integration with Jenkins**
10. Jenkins containerized using Docker
11. Installed required plugins (Docker, GitHub, NodeJS)
12. Configured Jenkins pipeline using `Jenkinsfile`
13. Triggered build/test steps on push
14. **Code Quality with SonarQube**
15. Set up SonarQube via Docker
16. Wrote `sonar-project.properties`
17. Scanned project from Jenkins and manually
18. **Helm Chart Creation**
19. Defined Helm charts for deployment and service
20. Set `imagePullPolicy: Never` for local testing
21. Configured values in `values.yaml`
22. **Kubernetes Deployment**
23. Deployed via Helm on Docker Desktop
24. Validated service via NodePort

25. **Monitoring Setup**

26. Initially attempted Prometheus and Grafana

27. Due to local issues, switched to Netdata (via DaemonSet)

28. AWS CloudWatch integrated during EKS deployment

29. **Cloud Deployment via AWS**

30. Created EKS cluster using eksctl

31. Pushed image to AWS ECR
32. Configured kubeconfig context with aws eks

33. Deployed app via kubectl apply and Helm

34. **Security Implementation**

35. Configured SSL certificates (self-signed) in Nginx

36. Added security headers (X-Content-Type, X-Frame-Options)

37. **Selenium UI Testing**

38. Developed test_app.py using Selenium and Pytest

39. Tested title, heading, navbar, and buttons
40. Captured screenshots on failure

41. Generated coverage reports

42. **Project Management with Jira**

43. Tracked tasks, sprints, and progress

44. Used Jira boards for visualizing development workflow

---

5. Project Structure (VS Code)

```
DevSecOps_Project/
├── public/
├── src/
├── .gitignore
├── Dockerfile
├── Jenkins/
│   ├── Jenkinsfile
│   └── test_app_pytest.py
├── nginx-service.yaml
├── nginx-deployment.yaml
├── sonar-project.properties
├── test_app.py
├── values.yaml
├── chart/
│   └── templates/
│       ├── deployment.yaml
│       └── service.yaml
├── report.html
└── README.md
```

---

6. Challenges and Solutions

Challenge	Solution
Prometheus setup failure	Switched to Netdata and CloudWatch
Helm install error (file size)	Used `.helmignore` to exclude large files
AWS CLI auth issue	Installed AWS CLI and configured IAM roles
Docker image not pulled	Ensured correct tags and set `imagePullPolicy: Never`
Kubernetes TLS timeout	Restarted Docker Desktop and kubelet

---

7. Results

- Successful CI/CD integration using Jenkins
- Static analysis integrated with SonarQube
- Dockerized build verified with Nginx
- Deployed via Helm to Kubernetes
- Monitored via Netdata (local) and CloudWatch (AWS)
- Selenium test suite passed for UI workflows

- SSL enabled and security headers configured
- Tasks tracked using Jira board and issue logs

---

8. How to Run Locally

```
git clone https://github.com/imakena2/devsecopsproject.git
cd devsecopsproject
npm install
npm run dev
```

To build and run Docker:

```
docker build -t devsecops-nginx .
docker run -p 8080:80 devsecops-nginx
```

To deploy via Helm:

```
helm install devsecops ./chart --set image.repository=devsecops-
nginx,image.tag=latest
```

To run tests:

```
pytest test_app.py
```

---

9. Conclusion

This project provided a comprehensive experience in implementing DevSecOps principles. Each phase from development to deployment, security, testing, and monitoring was addressed, offering a strong foundation for modern software delivery pipelines. The use of both local and cloud environments illustrated the flexibility and power of containerized DevOps workflows.

---

10. Author

Idah Makena Ncooro\ GitHub: imakena2

---

11. License

MIT License

Appendices

Appendix A: Phase 1 - Environment & Tools Setup

- Installed VS Code, Git, Docker, Helm, and AWS CLI

https://github.com/imakena2/DevSecops_Project/blob/main/Snapshots/1file%20structure.JPG

- Configured GitHub repo and local repo syncing

https://github.com/imakena2/DevSecops_Project

- Initialized Jira for issue tracking

https://github.com/imakena2/DevSecops_Project/blob/main/Snapshots/Jira%20Setup.JPG

Appendix B: Phase 2 - App Development & Containerization

- Developed frontend using React + Vite
- Created Dockerfile with Nginx serving static files

https://github.com/imakena2/DevSecops_Project/blob/main/Snapshots/nginx%20prometheus%20container.JPG
https://github.com/imakena2/DevSecops_Project/blob/main/Snapshots/image%20running.JPG

Appendix C: Phase 3 - CI/CD with Jenkins

- Dockerized Jenkins instance

https://github.com/imakena2/DevSecops_Project/blob/main/Snapshots/Jenkins%20Build.JPG

- Configured Jenkinsfile with build/test/deploy steps

https://github.com/imakena2/DevSecops_Project/blob/main/Snapshots/Jenkins%20Run.JPG
https://github.com/imakena2/DevSecops_Project/blob/main/Snapshots/Docker%20image.JPG
https://github.com/imakena2/DevSecops_Project/blob/main/Snapshots/sonarqube%20pass%20on%20Jenkins.JPG

Appendix D: Phase 4 - Code Quality & Testing

- Integrated SonarQube for static analysis

https://github.com/imakena2/DevSecops_Project/blob/main/Snapshots/Sonarqube%20report.JPG

- Created test_app.py using Selenium and Pytest

https://github.com/imakena2/DevSecops_Project/blob/main/Snapshots/Selenium%20Report.pdf
https://github.com/imakena2/DevSecops_Project/blob/main/Snapshots/Pytest%20Report2.JPG

Appendix E: Phase 5 - Kubernetes Deployment

- Wrote Helm charts for deployment

  https://github.com/imakena2/DevSecops_Project/blob/main/Snapshots/Helm.JPG

- Validated application via kubectl get svc

https://github.com/imakena2/DevSecops_Project/blob/main/Snapshots/Kubernetes%20deployment%20and%20service%20have%20been%20successfully%20created%20on%20AWS%20EKS.JPG

Appendix F: Phase 6 - Monitoring

- Installed Netdata DaemonSet on Docker Desktop
- Integrated AWS CloudWatch for EKS cluster
- https://github.com/imakena2/DevSecops_Project/blob/main/Snapshots/cloudwatch%20metrics.JPG
- https://github.com/imakena2/DevSecops_Project/blob/main/Snapshots/netdata.JPG
- https://github.com/imakena2/DevSecops_Project/blob/main/Snapshots/monitoring%20with%20Kubernetes%201.JPG
- https://github.com/imakena2/DevSecops_Project/blob/main/Snapshots/prometheus%20metrics.JPG

Appendix G: Phase 7 - Security

- Applied SSL and security headers in Nginx config
- Verified secure deployment via HTTPS and browser headers

Vulnerability Scanner

https://github.com/imakena2/DevSecops_Project/blob/main/Snapshots/Trivy.docx

SAST (Static Application Security Testing)

https://github.com/imakena2/DevSecops_Project/blob/main/Snapshots/SonarQube%20Jenkins.JPG

Appendix H: Phase 8 - Project Management

- Used Jira for sprint planning, backlog, and issue tracking

https://github.com/imakena2/DevSecops_Project/blob/main/Snapshots/Jira.JPG

Appendix I: Others

https://github.com/imakena2/DevSecops_Project/blob/main/Snapshots/ECR.JPG
https://github.com/imakena2/DevSecops_Project/blob/main/Snapshots/EKS%20cluster%20up.JPG
https://github.com/imakena2/DevSecops_Project/blob/main/Snapshots/Grafana%20running.JPG

Deployed App Via AWS

https://github.com/imakena2/DevSecops_Project/blob/main/Snapshots/App%20Running%20after%20I%20Deployed%20the%20app%20via%20Kubernetes%20deployment.JPG
https://github.com/imakena2/DevSecops_Project/blob/main/Snapshots/AWS%20cluster.JPG