Step-by-Step Instructions

DevSecOps Pipeline Setup for Spring PetClinic

---

0. System Requirements & Tested on

System Requirements

- OS: Ubuntu 22.04
- Architecture: x86_64
- CPU: AMD Ryzen 7 5800X, 8 cores / 16 threads
- Memory: Minimum 8 GB RAM
- Docker: Version 20.10+
- Docker Compose: v2.0+
- Java: OpenJDK 17+
- Maven: 3.6+
- Git: v2.39
- Python: 3.8+
- Virtualization:
    - VMware Workstation 17.6.1 build-24319023
    - Used for hosting the production Linux VM (Ansible target)

Tested on

- AMD Ryzen 7 5800X, 16 threads, Ubuntu Desktop
- Virtualization via VMware Workstation

1. Environment Setup

1.1. Fork and Clone the Repository

```
git clone https://github.com/teyenc/spring-petclinic.git
cd spring-petclinic
```

1.2. Set Up Custom Docker Network

```
docker network create devsecops-network
```

1.3. Prepare Docker Compose File

Create a docker-compose.yml with the following services:

- Jenkins (volume mount + pre-installed plugins)
- SonarQube
- Prometheus
- Grafana
- OWASP ZAP

Ensure all services are attached to devsecops-network.

1.4. Start All Services and verify that they're running

```
docker compose up -d --build
docker ps
```

2. Jenkins Configuration

2.1. Access Jenkins [Screenshot 2]

- Visit: http://localhost:8080
- Login with:
    - **Username**: admin
    - **Password**: From /var/jenkins_home/secrets/initialAdminPassword
- Setup new password

2.2. Install Required Plugins

Install the following via **Manage Jenkins → Plugin Manager**:

- Blue Ocean
- Pipeline
- Git
- Ansible
- Prometheus
- HTML Publisher
- SonarQube Scanner
- OWASP Dependency-Check Plugin

- Prometheus Metrics

2.3. Create a Pipeline Job [Screenshot 11]

- Go to Jenkins Dashboard → **New Item**
- Name: {pipe-line-name}
- Type: Pipeline
- In **Pipeline script from SCM**:
    - **SCM**: Git
    - **Repo URL**: https://github.com/teyenc/spring-petclinic.git
    - **Branch**: */main

2.4. Enable SCM Polling [Screenshot 12]

- Under **Build Triggers**, check Poll SCM
- Set schedule to: H/5 * * * *

3. Jenkinsfile Pipeline Configuration

- Ensure the Jenkinsfile in the repo includes:
    - Checkout
    - Build with Maven
    - SonarQube Static Analysis
    - OWASP Dependency Check (optional)
    - Build Docker Image
    - Run OWASP ZAP Scan
    - Publish ZAP Report
    - Deploy to VM using Ansible
- Make sure the Jenkinsfile is committed at the **repo root**.

4. SonarQube Setup

4.1. Access SonarQube [Screenshot 3]

- Visit: http://localhost:9000
- Login: admin / admin

4.2. Generate Token

- Go to **My Account → Security**
- Generate a new token

4.3. Configure SonarQube in Jenkins

- Go to **Manage Jenkins → Configure System**
- Under **SonarQube servers**:
    - Name: SonarQube
    - Server URL: http://sonarqube:9000
    - Credentials: Add token (ID: sonar-token)

5. Prometheus & Grafana Setup

5.1. Prometheus [Screenshot 4]

- Mount prometheus.yml with scrape targets:
    - jenkins:8080

5.2. Grafana [Screenshot 5, 6]

- Visit: http://localhost:3000
- Login: admin / admin
- Add **Prometheus** as a data source: http://prometheus:9090
- **Import Dashboards:**
    - Jenkins: ID 9964
    - SonarQube: ID 14627

6. OWASP ZAP Configuration [Screenshot 7]

- ZAP scan runs inside pipeline using zap-baseline.py Docker image
- HTML report is generated at: zap-report/zap-report.html
- Use publishHTML plugin to expose report:

```
publishHTML(target: [
  allowMissing: true,
  alwaysLinkToLastBuild: true,
  keepAll: true,
```

```
    reportDir: 'zap-report',
    reportFiles: 'zap-report.html',
    reportName: 'ZAP Security Report'
  ])
```

- **Access the Report:**
  - In Jenkins sidebar, navigate to **Build # → ZAP Security Report**

7. Ansible Deployment to Production VM

7.1. Prepare the VM

- Provision a Linux VM
- Install **Docker**
- Create a Dockerfile and open port 8080

7.2. Create Playbook

Create deploy/playbook.yml to:

- Pull latest image tagged with Jenkins build number
- Run spring-petclinic on port 8080

7.3. Jenkins Integration

- Use:

```
ansiblePlaybook(
  playbook: 'deploy/playbook.yml',
  inventory: 'deploy/inventory',
  extraVars: [
    build_number: env.BUILD_NUMBER
  ]
)
```

8. Trigger the Pipeline

8.1. Make a Code Change [Screenshot 9, 10]

```
echo "<h2>Hello World :) </h2>" >>
src/main/resources/templates/welcome.html
git add .
git commit -m "Trigger pipeline with update"
git push
```

8.2. Wait for Build [Screenshot 13-17]

- Jenkins will automatically:
    - Build
    - Analyze
    - Scan
    - Deploy
- After a successful build, access the web application to explore the updated features, and visit the monitoring services (Grafana, Prometheus, etc.) to review reports and observe system metrics.

9. Links

9.1 GitHub link

https://github.com/teyenc/spring-petclinic

9.2 DEMO Video Link
https://drive.google.com/file/d/10NCTHgAY3GwZZpWgIKOuvI5T7i4GkKJQ/view?usp=share_link