**Complete DevSecOps Guide: Secure Your Python App Using Docker!**

**Description:**

In this step-by-step **DevSecOps tutorial**, you'll learn how to **secure a Python Flask application** using **Docker, Docker Compose, and security tools**. We’ll go through **building, scanning, and securing your app** to follow best DevSecOps practices.

**Prerequisites**

1. Jenkins Installed on System.
2. **Git** Installed on System.
3. **Docker** Installed on System.

**Step 1: Set Up Your Environment**

1. Download Install **SonarQube** and **SonarScanner** from <https://github.com/rajandubey/DevSecOps_Project>
   1. Extract and Run SonarQube Service (sonarqube-25.3.0.104237\sonarqube-25.3.0.104237\bin\windows-x86-64\StartSonar.bat)
   2. Go to browser and hit <http://localhost:9000/>.
   3. Login with **admin / admin** (default credentials).
   4. Click on your **Profile (top-right corner)** → **My Account** → **Security**.
   5. Under **Generate Tokens**, enter a name (e.g., JenkinsToken).
   6. Click **Generate** and copy the **token**.
   7. Set the SONAR\_TOKEN Environment Variable
   8. Extract and Add SonarScanner to Windows System PATH (e.g. C:\sonar-scanner\bin)
   9. Configure SonarScanner to Use the Token
      1. Open the **SonarScanner configuration file (**C:\sonar-scanner\conf\sonar-scanner.properties)
      2. sonar.token={your-generated-token}
   10. Verify SonarScanner Authentication (sonar-scanner -Dsonar.projectKey= **squ\_94844d41cfa7d84e19f88b7ac4c06645c1e5c1d2** -Dsonar.sources=. -Dsonar.host.url=http://localhost:9000)
2. Install **OWASP Dependency-Check CLI** from<https://github.com/jeremylong/DependencyCheck/releases>
   1. Add to Windows Environment Variables Path (e.g. C:\dependency-check\bin)
   2. Verify Installation (dependency-check.bat --version)
3. Install **Trivy** for container scanning from <https://github.com/aquasecurity/trivy/releases>
   1. Add Trivy to System PATH (e.g. C:\Tools\Trivy)
   2. Verify Installation (trivy –version)

**Step 2: Create a Simple Web Application**

1. Write a basic web app in **Python (Flask)** **and** push it to **GitHub.**

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route("/")

def home():

    return "Hello, DevSecOps!"

if \_\_name\_\_ == "\_\_main\_\_":

    app.run(host="0.0.0.0", port=5000)

1. Create a Dockerfile:

FROM python:3.9

WORKDIR /app

COPY . /app

RUN pip install flask

CMD ["python", "app.py"]

1. Create a docker-compose.yml file:

version: "3.8"

services:

  flask-app:

    build: .

    ports:

      - "5000:5000"

    volumes:

      - .:/app

    restart: always

**Step 3: Set Up a Jenkins Pipeline**

1. Install Jenkins plugins:
   * Pipeline
   * SonarQube Scanner
   * OWASP Dependency-Check
   * Go to Jenkins->Manage Jenkins->System (Global Setting)
   * Enable “Environment variables” for SonarQube servers.
   * Restart Jenkins
2. Create a new pipeline in Jenkins and add the following steps:

pipeline {

    agent any

    stages {

        stage('Checkout Code') {

            steps {

                git branch: 'main', url: 'https://github.com/rajandubey/DevSecOps\_Project.git'

            }

        }

        stage('Static Code Analysis') {

            steps {

                bat 'sonar-scanner -Dsonar.projectKey=squ\_717ea518622ef86dcf85fbd1c3861ec44f4684d2 -Dsonar.sources=. -Dsonar.host.url=http://localhost:9000'

            }

        }

        stage('Dependency Scanning') {

            steps {

                bat 'dependency-check.bat --project DevSecOpsDemo --scan .'

            }

        }

        stage('Build and Scan Image') {

            steps {

                bat 'docker build -t myapp .'

                bat 'trivy image myapp'

            }

        }

        stage('Deploy') {

            steps {

                bat 'docker-compose up -d'

            }

        }

    }

}

1. Build Pipeline.
2. Monitor logs for errors
3. If errors occur, check:
   * Is **SonarQube running?**
   * Is **Trivy installed & working?**
   * Are **Docker services running?**
4. Verify Results
   * Check **SonarQube Reports** → <http://localhost:9000>
   * Check **Dependency Scanning Report** in the pipeline logs.
   * Check **Trivy Scan Report** in the logs.
   * Verify the **application is running.**
     + <http://localhost:5000>
5. Let’s quickly recap what we covered:
   * **DevSecOps** and why security is essential in CI/CD.
   * Set up **Jenkins** and integrated security tools like **SonarQube, Trivy, OWASP Dependency-Check** into our pipeline.
   * **Static Code Analysis** using **SonarQube** to detect code vulnerabilities.
   * **Vulnerable dependencies** with **OWASP Dependency-Check**.
   * **Scanned a Docker image** for security issues using **Trivy**.