Snyk Security Scan and Docker Integration

Overview

This document provides a step-by-step guide to integrating **Snyk Security Scans** with **GitHub Actions** for security vulnerability detection in the **Wednesday Adventures** project. The integration ensures that every code change is scanned for security vulnerabilities before being built and deployed as Docker images.

Key Concepts

What is Snyk?

Snyk is a developer-first security tool that identifies and fixes vulnerabilities in open-source dependencies, container images, and infrastructure as code.

Why Use Snyk in CI/CD?

- **Automated Security Scans**: Ensures that vulnerabilities are detected early in the development lifecycle.
- **Continuous Monitoring**: Tracks security issues in dependencies over time.
- **Seamless GitHub Integration**: Automatically scans repositories for security vulnerabilities.
- **Docker Image Security**: Scans containerized applications for vulnerabilities before deployment.

Requirements

Prerequisites

- **Snyk Account** (Sign up here)
- **GitHub Repository** configured for CI/CD

GitHub Secrets

Secret Name	Description

`SNYK_TOKEN`	Authentication token for Snyk
`DOCKER_USERNAME`	Docker Hub username
`DOCKER_PASSWORD`	Docker Hub password

Steps to Set Up Snyk Security Integration

1. Configure Snyk Project

- Create a Snyk project and obtain the **API Token**.
- Store the **SNYK_TOKEN** in **GitHub Secrets**.

2. GitHub Actions Workflow

This workflow triggers Snyk security scans on **pushes to key branches** (`l00187927-WA-Jira18-CIPipeline`, `main`, `develop`) and **builds Docker images**.

File Location: `.github/workflows/snyk-security-docker.yml`

```yaml
name: Snyk Security Scan and Docker

on:
  push:
    branches:
      - l00187927-WA-Jira18-CIPipeline
      - main
      - develop

jobs:
  build-test-security:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout code
      uses: actions/checkout@v3

    - name: Set up Node.js
      uses: actions/setup-node@v3
```

```
      with:
        node-version: 20
        cache: 'npm'

    - name: Install backend dependencies
      working-directory: ./project-backend
      run: npm ci

    - name: Install frontend dependencies
      working-directory: ./frontend
      run: npm ci

    - name: Install Snyk CLI
      run: npm install -g snyk

    - name: Validate Snyk API Token
      run: |
        if [ -z "${{ secrets.SNYK_TOKEN }}" ]; then
          echo "ERROR: SNYK_TOKEN is not set or is empty!"
          exit 1
        fi

    - name: Run Snyk Security Scan
      env:
        SNYK_TOKEN: ${{ secrets.SNYK_TOKEN }}
      run: snyk test --all-projects --severity-threshold=high || echo
"Snyk scan completed with issues"

    - name: Monitor vulnerabilities in Snyk
      env:
        SNYK_TOKEN: ${{ secrets.SNYK_TOKEN }}
      run: snyk monitor --all-projects || echo "Snyk monitor completed
with issues"

    - name: Build Backend Docker Image
      run: docker build -t ${{ secrets.DOCKER_USERNAME
}}/wednesday-adventures-backend:latest -f Dockerfile.backend .

    - name: Build Frontend Docker Image
```

```
      run: docker build -t ${{ secrets.DOCKER_USERNAME
}}/wednesday-adventures-frontend:latest -f Dockerfile.frontend .

    - name: Verify Docker Images
      run: |
        docker images | grep ${{ secrets.DOCKER_USERNAME
}}/wednesday-adventures-backend || exit 1
        docker images | grep ${{ secrets.DOCKER_USERNAME
}}/wednesday-adventures-frontend || exit 1

    - name: Log in to Docker Hub
      run: echo "${{ secrets.DOCKER_PASSWORD }}" | docker login -u "${{
secrets.DOCKER_USERNAME }}" --password-stdin

    - name: Push Backend Docker Image
      run: docker push ${{ secrets.DOCKER_USERNAME
}}/wednesday-adventures-backend:latest

    - name: Push Frontend Docker Image
      run: docker push ${{ secrets.DOCKER_USERNAME
}}/wednesday-adventures-frontend:latest

    - name: Verify Images on Docker Hub
      run: |
        docker pull ${{ secrets.DOCKER_USERNAME
}}/wednesday-adventures-backend:latest
        docker pull ${{ secrets.DOCKER_USERNAME
}}/wednesday-adventures-frontend:latest

    - name: Snyk Container Scan for Backend
      env:
        SNYK_TOKEN: ${{ secrets.SNYK_TOKEN }}
      run: snyk container test ${{ secrets.DOCKER_USERNAME
}}/wednesday-adventures-backend:latest --severity-threshold=high || echo
"Snyk container scan completed with issues"

    - name: Snyk Container Scan for Frontend
      env:
        SNYK_TOKEN: ${{ secrets.SNYK_TOKEN }}
```

```
      run: snyk container test ${{ secrets.DOCKER_USERNAME
}}/wednesday-adventures-frontend:latest --severity-threshold=high || echo
"Snyk container scan completed with issues"
```

Workflow Steps

1. Checkout Code

- name: Checkout code
  uses: actions/checkout@v3

This step pulls the latest code from the repository.

2. Set Up Node.js Environment

- name: Set up Node.js
  uses: actions/setup-node@v3
  with:
    node-version: 20
    cache: 'npm'

It installs Node.js version 20 and enables npm caching to optimize dependency installation.

3. Install Backend Dependencies

- name: Install backend dependencies
  working-directory: ./project-backend
  run: npm ci

This command installs the dependencies for the backend application using `npm ci`.

4. Install Frontend Dependencies

- name: Install frontend dependencies
  working-directory: ./frontend
  run: npm ci

Similarly, it installs dependencies for the frontend application.

5. Install Snyk CLI

```
- name: Install Snyk CLI
  run: npm install -g snyk
```

Snyk CLI is installed globally to enable security scanning.

6. Validate Snyk API Token

```
- name: Validate Snyk API Token
  run: |
    if [ -z "${{ secrets.SNYK_TOKEN }}" ]; then
      echo "ERROR: SNYK_TOKEN is not set or is empty!"
      exit 1
    fi
```

Ensures that the `SNYK_TOKEN` secret is properly configured.

7. Run Snyk Security Scan

```
- name: Run Snyk Security Scan
  env:
    SNYK_TOKEN: ${{ secrets.SNYK_TOKEN }}
  run: snyk test --all-projects --severity-threshold=high || echo "Snyk scan completed with issues"
```

Performs a security scan and reports high-severity issues.

8. Monitor Vulnerabilities with Snyk

```
- name: Monitor vulnerabilities in Snyk
  env:
    SNYK_TOKEN: ${{ secrets.SNYK_TOKEN }}
  run: snyk monitor --all-projects || echo "Snyk monitor completed with issues"
```

Uploads dependency information to Snyk for continuous monitoring.

9. Build Backend Docker Image

```
- name: Build Backend Docker Image
  run: docker build -t ${{ secrets.DOCKER_USERNAME
}}/wednesday-adventures-backend:latest -f Dockerfile.backend .
```

Builds the backend Docker image using `Dockerfile.backend`.

10. Build Frontend Docker Image

```
- name: Build Frontend Docker Image
  run: docker build -t ${{ secrets.DOCKER_USERNAME
}}/wednesday-adventures-frontend:latest -f Dockerfile.frontend .
```

Builds the frontend Docker image using `Dockerfile.frontend`.

11. Verify Docker Images

```
- name: Verify Docker Images
  run: |
    docker images | grep ${{ secrets.DOCKER_USERNAME }}/wednesday-adventures-backend
|| exit 1
    docker images | grep ${{ secrets.DOCKER_USERNAME }}/wednesday-adventures-frontend
|| exit 1
```

Ensures that the Docker images have been built successfully.

12. Log in to Docker Hub

```
- name: Log in to Docker Hub
  run: echo "${{ secrets.DOCKER_PASSWORD }}" | docker login -u "${{
secrets.DOCKER_USERNAME }}" --password-stdin
```

Logs into Docker Hub using credentials stored in GitHub secrets.

13. Push Backend Docker Image to Docker Hub

```
- name: Push Backend Docker Image
  run: docker push ${{ secrets.DOCKER_USERNAME }}/wednesday-adventures-backend:latest
```

Pushes the backend image to Docker Hub.

14. Push Frontend Docker Image to Docker Hub

```
- name: Push Frontend Docker Image
  run: docker push ${{ secrets.DOCKER_USERNAME }}/wednesday-adventures-frontend:latest
```

Pushes the frontend image to Docker Hub.

15. Verify Images on Docker Hub

```
- name: Verify Images on Docker Hub
  run: |
    docker pull ${{ secrets.DOCKER_USERNAME }}/wednesday-adventures-backend:latest
    docker pull ${{ secrets.DOCKER_USERNAME }}/wednesday-adventures-frontend:latest
```

Pulls the images from Docker Hub to ensure they were pushed successfully.

Explanation of Workflow Steps

Step	Description
Checkout Code	Retrieves the latest repository code.
Set up Node.js	Configures Node.js (version 20) with npm caching.
Install Dependencies	Runs `npm ci` for backend and frontend to ensure consistency.
Validate Snyk API Token	Ensures the SNYK_TOKEN is available before scanning.
Run Snyk Security Scan	Checks for vulnerabilities in dependencies and fails on `high` severity issues.
Monitor Snyk Vulnerabilities	Tracks project vulnerabilities over time in Snyk.
Build Backend Docker Image	Creates a containerized image for the backend service.
Build Frontend Docker Image	Creates a containerized image for the frontend service.
Verify Docker Images	Ensures that Docker images were built successfully.
Log in to Docker Hub	Authenticates with Docker Hub using stored credentials.
Push Backend Docker Image	Uploads the backend image to Docker Hub.
Push Frontend Docker Image	Uploads the frontend image to Docker Hub.
Verify Images on Docker Hub	Ensures that pushed images can be pulled from Docker Hub.

How to Review Security Reports

1. **Snyk Dashboard**: Visit Snyk and navigate to your project.
2. **GitHub Pull Requests**: Check Snyk annotations in GitHub PRs for security feedback.
3. **Snyk CLI**: Run `snyk test` locally to scan dependencies before pushing code.

Troubleshooting

Issue	Solution
Snyk Scan Fails	Ensure `SNYK_TOKEN` is correctly set in GitHub Secrets.
Docker Image Not Found	Verify `Dockerfile.backend` and `Dockerfile.frontend` exist in the repository.
GitHub Secrets Missing	Ensure `DOCKER_USERNAME`, `DOCKER_PASSWORD`, and `SNYK_TOKEN` are set in repository secrets.
Login to Docker Hub Fails	Check if the credentials are correct and Docker Hub is accessible.

Conclusion

This workflow ensures that every code change undergoes an automated security scan and is built into Docker images before deployment, enhancing security and reliability in the **Wednesday Adventures** project.