National College of Ireland

Configuration Manual

MSc Research Project
MSc Cloud Computing

Siva Suriya Kandasamy
Student ID: 23341963

School of Computing
National College of Ireland

Supervisor: Sean Heeney

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Siva Suriya Kandasamy
Student ID:	23341963
Programme:	Masters in Cloud Computing **Year:** 2024 - 2025
Module:	MSc Research Project
Lecturer:	Sean Heeney
Submission Due Date:	11/08/2025
Project Title:	Practical Implementation of Zero Trust Security in DevSecOps Pipelines
Word Count:	924 **Page Count:** 15

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: 

Date: 11/08/2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Siva Suriya Kandasamy
23341963

1 Environment Setup

1) Log in to your AWS Management Console.
2) Navigate to the EC2 instance and launch the instance.
3) Choose the following configurations:
    a. AMI: Ubuntu Server 22.04 LTS (64-bit)
    b. Instance type: t3.medium (2 vCPU, 4 GB RAM)
    c. Storage: 30 GB GP2 EBS volume
    d. Security Group: Security Group enabled with port 30673 for the Kubernetes NodePort range
4) Connect to the instance.
5) Install K3 (light-weight) Kubernetes and verify if the nodes are ready

```
curl -sfL https://get.k3s.io | sh -
sudo kubectl get nodes --kubeconfig=/etc/rancher/k3s/k3s.yaml
```

```
voclabs:~/environment (main) $ sudo kubectl get nodes --kubeconfig=/etc/rancher/k3s/k3s.yaml
NAME                            STATUS   ROLES                  AGE   VERSION
ip-172-31-24-157.ec2.internal   Ready    control-plane,master   44d   v1.32.5+k3s1
```

6) To avoid image-pull delays, an EBS volume has been created, and the below images are pre-loaded.
    a. CoreDNS
    b. metric-server
    c. local-path-provisioner
7) Check if the core services / pods are running correctly.

```
voclabs:~/environment/istio-1.21.1 (main) $ sudo kubectl get pods -n kube-system --kubeconfig=/etc/rancher/k3s/k3s.yaml
NAME                                      READY   STATUS    RESTARTS         AGE
coredns-5c9b6b557c-9d7wf                  1/1     Running   19 (8m20s ago)   9d
local-path-provisioner-5c88657fd9-tn5z4   1/1     Running   18 (8m20s ago)   9d
metrics-server-8479b9b579-dbmfk           1/1     Running   19 (8m20s ago)   9d
traefik-6ddc45c878-fhlw5                  1/1     Running   57 (8m20s ago)   19d
```

2 Vulnerable Application Deployment (OWASP Juice Shop)

1) Now, we need to create the Kubernetes deployment. We use OWASP Juice Shop as the vulnerable web application and we are deploying it in default namespace.

```
  sudo kubectl create deployment juice-shop \
--image=bkimminich/juice-shop \
--kubeconfig=/etc/rancher/k3s/k3s.yaml
```

2) Now, we need to set the application that it can be accessed externally via NodePort.

```
sudo kubectl expose deployment juice-shop \
--type=NodePort \
--port=3000 \
--name=juice-shop-nodeport \
--kubeconfig=/etc/rancher/k3s/k3s.yaml
```

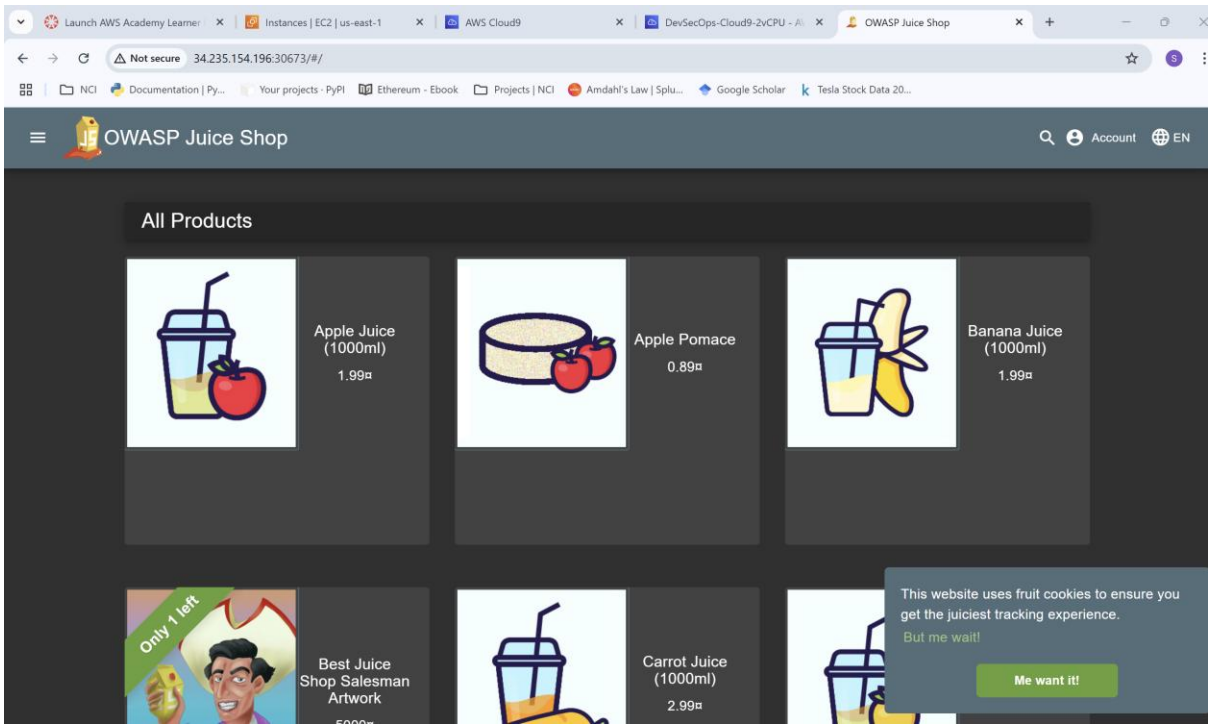
3) Check the Service if its enabled correctly.

```
voclabs:~/environment/istio-1.21.1 (main) $ sudo kubectl get svc juice-shop-nodeport \
>   --kubeconfig=/etc/rancher/k3s/k3s.yaml
NAME                 TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
juice-shop-nodeport  NodePort   10.43.82.162    <none>        3000:30673/TCP   5d19h
```

4) Check the Juice shop pod, if its in running state.

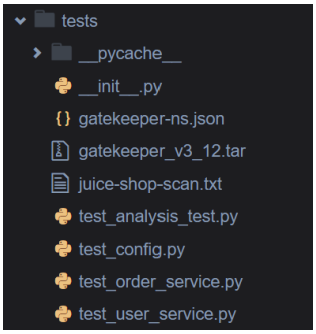
```
voclabs:~/environment/istio-1.21.1 (main) $ sudo kubectl get pods -n default --kubeconfig=/etc/rancher/k3s/k3s.yaml
NAME                          READY   STATUS    RESTARTS   AGE
juice-shop-8648fb9b5b-5x8qd   1/1     Running   0          13m
```

5) Check the OWASP Juice shop browser.



3 Sonar Cloud

1) SonarCloud is being used for testing the code quality and a good security practice.
2) SonarCloud is linked with the GitHub repo account, so that it will be triggered when the main branch in the repo gets updated.
3) In this research, it's being implemented by testing four python files to detect the correct errors and expected security issues to align with the research.
4) Below are the four .py files used in our project.
    a. test_analysis.py
    b. test_config.py
    c. test_order_service.py
    d. test_user_service.py



5) To run this test, Integrated this process with the GitHub actions.
6) Created a yaml file named build.yml which runs the SonarCloud test via GitHub actions.
7) This will be triggered when the main branch is pushed.

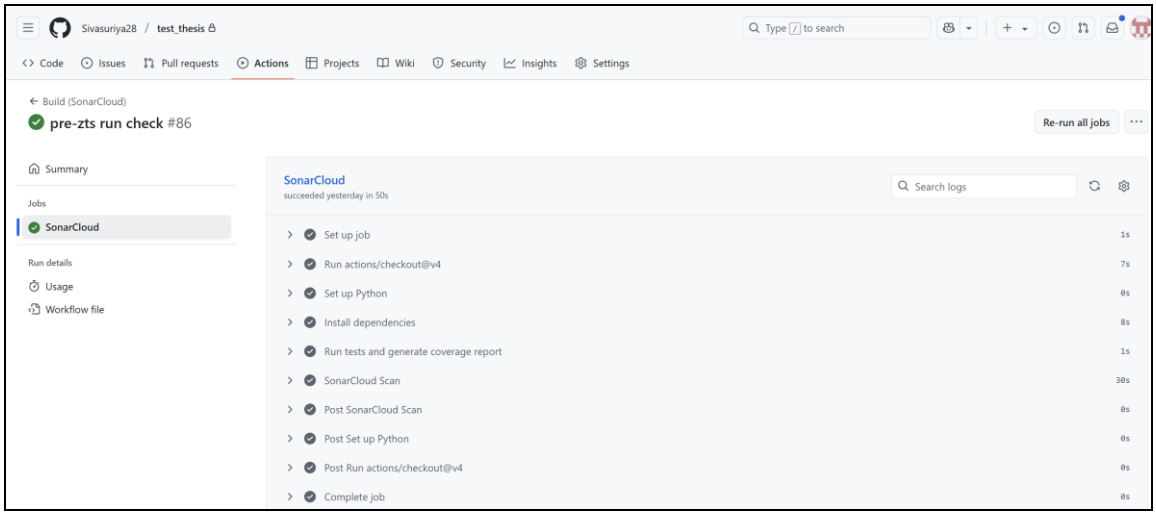
```
voclabs:~/environment (main) $ git push origin main
```

8) Below is the snippet of build.yml file.

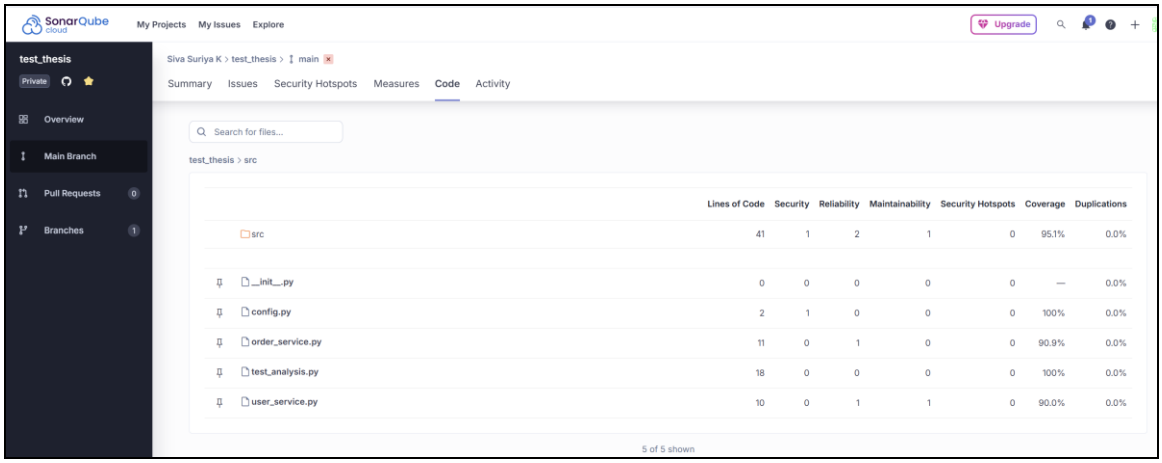
build.yml

```yaml
name: Build (SonarCloud)

on:
  push:
    branches:
      - main
  pull_request:
    types: [opened, synchronize, reopened]

jobs:
  sonarqube:
    name: SonarCloud
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v4
        with:
          fetch-depth: 0

      - name: Set up Python
        uses: actions/setup-python@v5
        with:
          python-version: '3.9'

      - name: Install dependencies
        run: |
          pip install -r requirements.txt
          pip install pytest pytest-cov

      - name: Run tests and generate coverage report
        run: |
          PYTHONPATH=. pytest --cov=src --cov-report=xml

      - name: SonarCloud Scan
        uses: SonarSource/sonarqube-scan-action@v5
        with:
          projectBaseDir: .
        env:
          SONAR_TOKEN: ${{ secrets.SONAR_TOKEN }}
```

9) Once this file is triggered, build stage will be ran in the GitHub Actions.



10) Sonar Cloud dashboard will be updated with the security checks done.



4 Pre-ZTS Security Scan

1) In Pre-ZTS Security Scan, the OWASP Juice Shop application is being scanned by using two scanners namely Trivy (SAST) and OWASP Zap (DAST).
2) pre-zts.yml file has been created which has both Trivy installation and OWASP zap running in a GitHub hosted environment.
3) By this way, manual installation is not required.
4) Below is the screenshot of pre-zts.yml file with Trivy scan.

```yaml
jobs:
  trivy_scan:
    runs-on: ubuntu-latest
    name: Trivy Complete Vulnerability Scan

    steps:
      - name: Checkout code
        uses: actions/checkout@v4

      - name: Install Trivy
        run: |
          sudo apt-get update
          sudo apt-get install wget apt-transport-https gnupg lsb-release -y
          wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | sudo apt-key add -
          echo "deb https://aquasecurity.github.io/trivy-repo/deb $(lsb_release -sc) main" | sudo tee /etc/apt/sources.list.d/trivy.list
          sudo apt-get update
          sudo apt-get install trivy -y

      - name: Prepare Trivy Reports Directory
        run: mkdir -p trivy-reports

      - name: Trivy FS Scan
        run: trivy fs . --format table --output trivy-reports/fs-scan.txt

      - name: Trivy requirements.txt Scan
        run: |
          if [ -f requirements.txt ]; then
            trivy fs --security-checks vuln \
              --file-patterns requirements:requirements.txt \
              --format table \
              --output trivy-reports/requirements-scan.txt .
          else
            echo "No requirements.txt found. Skipping scan."
          fi

      - name: Trivy Config Scan
        run: trivy config . --format table --output trivy-reports/config-scan.txt

      - name: Trivy Image Scan
        run: trivy image bkimminich/juice-shop --format table --output trivy-reports/image-scan.txt --ignore-unfixed --severity CRITICAL,HIGH

      - name: Upload Trivy Reports
        uses: actions/upload-artifact@v4
        with:
          name: trivy-scan-report
          path: trivy-reports/
```

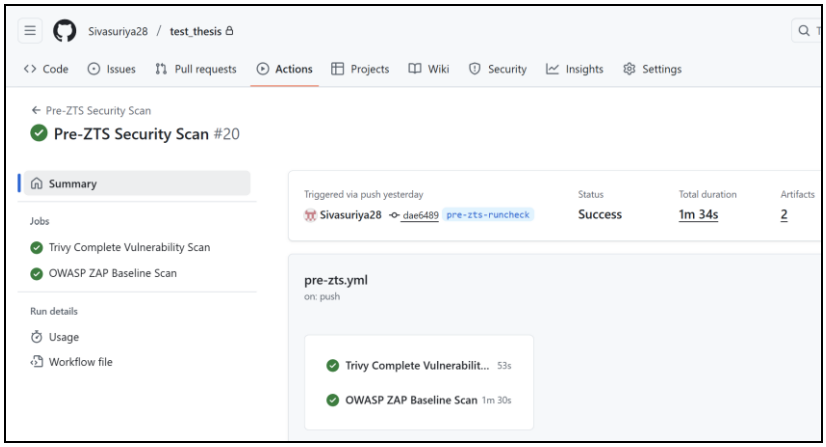
5) Below is the screenshot of pre-zts.yml file with OWASP ZAP.

```yaml
zap_scan:
  runs-on: ubuntu-latest
  name: OWASP ZAP Baseline Scan

  steps:
    - name: Checkout code
      uses: actions/checkout@v4

    - name: Prepare ZAP Reports
      run: mkdir -p zap-reports

    - name: Run OWASP ZAP Baseline Scan
      run: |
        docker run --rm \
          -u root \
          -v ${{ github.workspace }}/zap-reports:/zap/wrk \
          ghcr.io/zaproxy/zaproxy:stable \
          zap-baseline.py \
          -t http://54.147.191.178:30673 \
          -J zap-report.json \
          -w zap-report.md \
          -r zap-report.html
      continue-on-error: true

    - name: Upload ZAP Report
      uses: actions/upload-artifact@v4
      with:
        name: zap-scan-report
        path: zap-reports/
```

6) By running this file, Both Trivy and ZAP will be installed and ran from the GitHub Actions engine, and the scan reports will be generated.
7) A single repo is maintained for both Pre-ZTS and Post-ZTS pipeline, so tags are used to push the file to GitHub Actions.
8) “pre-zts-*” tag is used to push and run the pre-zts CI/CD pipeline.
9) Reports generated for Trivy scans:
    a. fs-scan.txt
    b. requirements-scan.txt
    c. config-scan.txt

d. image-scan.txt

10) Reports generated for ZAP scans:
    a. zap-report.html
    b. zap-report.md
    c. zap-report.json
11) Snippet of Pre-ZTS stage in GitHub Actions:



12) After completion of pre-zts scan, artifacts will be downloaded and used further for comparison.

5 OPA Gatekeeper Setup and Policy Enforcement

1) Install OPA Gatekeeper

```
sudo kubectl apply -f https://raw.githubusercontent.com/open-policy-agent/gatekeeper/release-3.15/deploy/gatekeeper.yaml \
--kubeconfig=/etc/rancher/k3s/k3s.yaml
```

```
voclabs:~/environment (main) $ sudo kubectl apply -f https://raw.githubusercontent.com/open-policy-agent/gatekeeper/release-3.13/deploy/gatekeeper.yaml --kubeconfig=/etc/rancher/k
3s/k3s.yaml
namespace/gatekeeper-system created
resourcequota/gatekeeper-critical-pods created
customresourcedefinition.apiextensions.k8s.io/assign.mutations.gatekeeper.sh created
customresourcedefinition.apiextensions.k8s.io/assignimage.mutations.gatekeeper.sh created
customresourcedefinition.apiextensions.k8s.io/assignmetadata.mutations.gatekeeper.sh created
customresourcedefinition.apiextensions.k8s.io/configs.config.gatekeeper.sh created
customresourcedefinition.apiextensions.k8s.io/constraintpodstatuses.status.gatekeeper.sh created
customresourcedefinition.apiextensions.k8s.io/constrainttemplatepodstatuses.status.gatekeeper.sh created
customresourcedefinition.apiextensions.k8s.io/constrainttemplates.templates.gatekeeper.sh created
customresourcedefinition.apiextensions.k8s.io/expansiontemplate.expansion.gatekeeper.sh created
customresourcedefinition.apiextensions.k8s.io/expansiontemplatepodstatuses.status.gatekeeper.sh created
customresourcedefinition.apiextensions.k8s.io/modifyset.mutations.gatekeeper.sh created
customresourcedefinition.apiextensions.k8s.io/mutatorpodstatuses.status.gatekeeper.sh created
customresourcedefinition.apiextensions.k8s.io/providers.externaldata.gatekeeper.sh created
serviceaccount/gatekeeper-admin created
role.rbac.authorization.k8s.io/gatekeeper-manager-role created
clusterrole.rbac.authorization.k8s.io/gatekeeper-manager-role created
rolebinding.rbac.authorization.k8s.io/gatekeeper-manager-rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/gatekeeper-manager-rolebinding created
secret/gatekeeper-webhook-server-cert created
service/gatekeeper-webhook-service created
deployment.apps/gatekeeper-audit created
deployment.apps/gatekeeper-controller-manager created
poddisruptionbudget.policy/gatekeeper-controller-manager created
mutatingwebhookconfiguration.admissionregistration.k8s.io/gatekeeper-mutating-webhook-configuration created
validatingwebhookconfiguration.admissionregistration.k8s.io/gatekeeper-validating-webhook-configuration created
```

2) Verify Gatekeeper Installation.

```
voclabs:~/environment (main) $ sudo kubectl get pods -n gatekeeper-system --kubeconfig=/etc/rancher/k3s/k3s.yaml
NAME                                             READY   STATUS    RESTARTS      AGE
gatekeeper-audit-5f49b8976b-h88bp                1/1     Running   2 (67s ago)   70s
gatekeeper-controller-manager-5688d8945b-68rrh   1/1     Running   0             70s
gatekeeper-controller-manager-5688d8945b-hx8t4   1/1     Running   0             70s
gatekeeper-controller-manager-5688d8945b-zvcm7   1/1     Running   0             70s
```

3) Constraint templates have been used to enforce policies.
4) Two yaml files has been used to enforce the policies. K8srequiredlabels.yaml and require-app-label.yaml

```
voclabs:~/environment (main) $ sudo kubectl apply -f opa/k8srequiredlabels.yaml --kubeconfig=/etc/rancher/k3s/k3s.yaml
constrainttemplate.templates.gatekeeper.sh/k8srequiredlabels created
voclabs:~/environment (main) $ sudo kubectl apply -f opa/require-app-label.yaml --kubeconfig=/etc/rancher/k3s/k3s.yaml
k8srequiredlabels.constraints.gatekeeper.sh/must-have-app-label created
```

k8srequiredlabels.yam ×

```yaml
apiVersion: templates.gatekeeper.sh/v1beta1
kind: ConstraintTemplate
metadata:
  name: k8srequiredlabels
spec:
  crd:
    spec:
      names:
        kind: K8sRequiredLabels
  targets:
    - target: admission.k8s.gatekeeper.sh
      rego: |
        package k8srequiredlabels

        violation[{"msg": msg}] {
          provided := {label | input.review.object.metadata.labels[label]}
          required := {label | label := input.parameters.labels[_]}
          missing := required - provided
          count(missing) > 0
          msg := sprintf("Missing required labels: %v", [missing])
        }
```

require-app-label.yaml ×

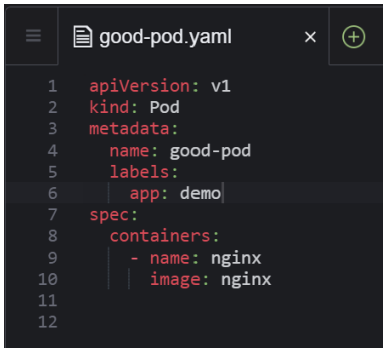
```yaml
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sRequiredLabels
metadata:
  name: must-have-app-label
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
  parameters:
    labels: ["app"]
```

5) Ensure the policy is enforced correctly.

```
voclabs:~/environment (main) $ sudo kubectl get constraints --all-namespaces --kubeconfig=/etc/rancher/k3s/k3s.yaml
NAME                 ENFORCEMENT-ACTION   TOTAL-VIOLATIONS
must-have-app-label                       5
```

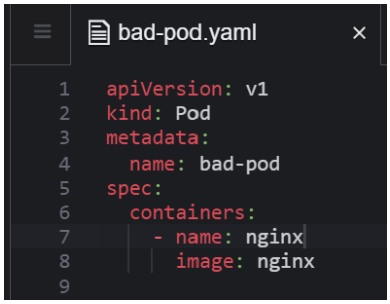
6) Now, if pod created without app label, it should throw an error and deny from creating a pod.
7) Created two yaml files named good-pod with app label and bad-pod which doesn't have the app label in it.

8) Snippets of good-pod and bad-pod yaml files:
9) Creation of new pod, when app label is present.




```
voclabs:~/environment/opa/testpods (main) $ sudo kubectl apply -f good-pod.yaml --kubeconfig=/etc/rancher/k3s/k3s.yaml
pod/good-pod created
```

10) Denial of pod creation, when app label is absent.




```
voclabs:~/environment/opa/testpods (main) $ sudo kubectl apply -f bad-pod.yaml --kubeconfig=/etc/rancher/k3s/k3s.yaml
Error from server (Forbidden): error when creating "bad-pod.yaml": admission webhook "validation.gatekeeper.sh" denied the request: [must-have-app-label] Missing required labels:
{"app"}
```

6 Istio Installation and AuthorizationPolicy

1) Install Istio.

```
curl -L https://istio.io/downloadIstio | ISTIO_VERSION=1.22.3 sh -
```

```
voclabs:~/environment/opa/testpods (main) $ curl -L https://istio.io/downloadIstio | ISTIO_VERSION=1.22.3 sh -
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   101  100   101    0     0    808      0 --:--:-- --:--:-- --:--:--   814
100  5124  100  5124    0     0  30246      0 --:--:-- --:--:-- --:--:-- 30246

Downloading istio-1.22.3 from https://github.com/istio/istio/releases/download/1.22.3/istio-1.22.3-linux-amd64.tar.gz ...

Istio 1.22.3 download complete!
```

```
voclabs:~/environment/opa/testpods (main) $ sudo istioctl install --set profile=demo -y --kubeconfig=/etc/rancher/k3s/k3s.yaml
WARNING: Istio 1.21.0 may be out of support (EOL) already: see https://istio.io/latest/docs/releases/supported-releases/ for supported releases
✓ Istio core installed
✓ Istiod installed
✓ Egress gateways installed
✓ Ingress gateways installed
✓ Installation complete
Made this installation the default for injection and validation.
```

2) Verify pods in istio-system.

```
voclabs:~/environment/opa/testpods (main) $ sudo kubectl get pods -n istio-system --kubeconfig=/etc/rancher/k3s/k3s.yaml
NAME                                   READY   STATUS    RESTARTS   AGE
istio-egressgateway-667d9ccd4c-pwnsq   1/1     Running   0          90s
istio-ingressgateway-55f758c489-4b2nd  1/1     Running   0          90s
istiod-5df7b97f8d-54vss                1/1     Running   0          97s
```

3) Namespace “zts-test” has been labeled for this Istio sidecar injection.
4) Confirm if sidecar injection is enabled (2/2).

```
voclabs:~/environment/opa/testpods (main) $ sudo kubectl get namespace zts-test --show-labels --kubeconfig=/etc/rancher/k3s/k3s.yaml
NAME       STATUS   AGE   LABELS
zts-test   Active   19d   istio-injection=enabled,kubernetes.io/metadata.name=zts-test
```

```
voclabs:~/environment (main) $ sudo kubectl get pods -n zts-test --kubeconfig=/etc/rancher/k3s/k3s.yaml
NAME                       READY   STATUS    RESTARTS   AGE
httpbin-56cc78c99f-mzm46   1/1     Running   0          108s
sleep-7b8dd56cd9-px9c4     1/1     Running   0          108s
```

```
voclabs:~/environment/opa/testpods (main) $ sudo kubectl get svc  -n zts-test --kubeconfig=/etc/rancher/k3s/k3s.yaml
NAME      TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
httpbin   ClusterIP   10.43.138.30    <none>        8000/TCP   13d
sleep     ClusterIP   10.43.231.156   <none>        80/TCP     13d
```

5) Ensure that currently no authorization policies are enabled.

```
voclabs:~/environment/opa/testpods (main) $ sudo kubectl get authorizationpolicy -n zts-test --kubeconfig=/etc/rancher/k3s/k3s.yaml
No resources found in zts-test namespace.
```

6) Two policies will be implemented in this process. First comes deny-all policy which denies every request and will get an RBAC denied output.
7) Policy deny-all is being applied.

deny-all.yaml

```yaml
apiVersion: security.istio.io/v1beta1
kind: AuthorizationPolicy
metadata:
  name: deny-all
  namespace: zts-test
spec:
  {}
```

```
voclabs:~/environment/istio-1.21.1/deployments (main) $ sudo kubectl apply -f deny-all.yaml --kubeconfig=/etc/rancher/k3s/k3s.yaml
authorizationpolicy.security.istio.io/deny-all created
```

8) When reached to the server, got an error message, such as RBAC access denied as mentioned in the policy.

```
voclabs:~/environment/istio-1.21.1/deployments (main) $ sudo kubectl exec -n zts-test  deploy/sleep -c sleep --kubeconfig=/etc/rancher/k3s/k3s.yaml -- curl -sS http://httpbin:8000
/ip
RBAC: access deniedvoclabs:~/environment/istio-1.21.1/deployments (main) $
```

9) Similarly, now a new policy is being created in which service to service communication is allowed only from sleep to httpbin service.
10) Applying the policy “allow-sleep-to-httpbin”.

allow-sleep-to-httpbin.y ×

```yaml
apiVersion: security.istio.io/v1beta1
kind: AuthorizationPolicy
metadata:
  name: allow-sleep-to-httpbin
  namespace: zts-test
spec:
  selector:
    matchLabels:
      app: httpbin
  rules:
  - from:
    - source:
        principals: ["cluster.local/ns/zts-test/sa/default"]
```

```
voclabs:~/environment/istio-1.21.1/deployments (main) $ sudo kubectl apply -f allow-sleep-to-httpbin.yaml --kubeconfig=/etc/rancher/k3s/k3s.yaml
authorizationpolicy.security.istio.io/allow-sleep-to-httpbin created
```

11) When reached the server, got the correct access and server has been passed.

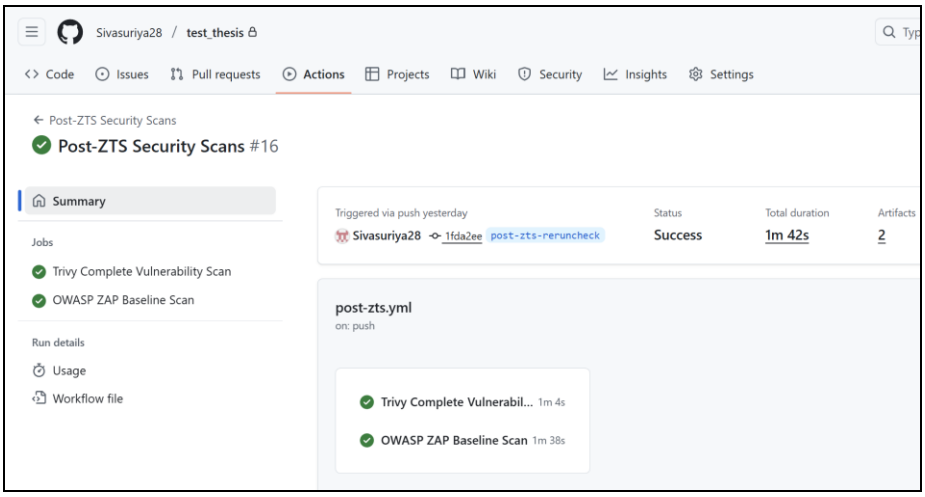
```
voclabs:~/environment/istio-1.21.1/deployments (main) $ sudo kubectl exec -n zts-test  deploy/sleep -c sleep --kubeconfig=/etc/rancher/k3s/k3s.yaml -- curl -sS http://httpbin:8000
/ip
{
  "origin": "127.0.0.6"
}
```

7 Post-ZTS Security Scan

1) After successful implementation of ZTS principles via OPA gatekeeper and Istio Service Mesh, again the Post-ZTS pipeline is being triggered to scan the application and get the reports.
2) A post-zts yaml file has been created and triggered which contains both Trivy and ZAP scans in it like Pre-ZTS pipeline.
3) The Post-ZTS pipeline is triggered by using git tag “post-zts-*”.

```
voclabs:~/environment (main) $ git push origin post-zts-*
```

4) Reports generated for Trivy scans:
    a. fs-scan.txt
    b. requirements-scan.txt
    c. config-scan.txt
    d. image-scan.txt
5) Reports generated for ZAP scans:
    a. zap-report.html
    b. zap-report.md
    c. zap-report.json
6) Snippet of Pre-ZTS stage in GitHub Actions:



7) After completion of pre-zts scan, artifacts will be downloaded and used further for comparison.

8 Generated Scan Reports

1) Trivy scan reports:

Name	Type	Compressed size	Password pr...	Size	Ratio	Date modified
config-scan	TXT File	16 KB	No	282 KB	95%	8/5/2025 2:31 PM
fs-scan	TXT File	3 KB	No	27 KB	90%	8/5/2025 2:31 PM
image-scan	TXT File	15 KB	No	591 KB	98%	8/5/2025 2:32 PM
requirements-scan	TXT File	3 KB	No	27 KB	90%	8/5/2025 2:31 PM

2) Snippet from the Trivy scan report:

```
200
201   istio-1.21.1/deployments/httpbin-deployment.yaml (kubernetes)
202   ============================================================
203   Tests: 125 (SUCCESSES: 106, FAILURES: 19)
204   Failures: 19 (UNKNOWN: 0, LOW: 10, MEDIUM: 6, HIGH: 3, CRITICAL: 0)
```

ZAP scan Reports:

zap	Yaml Source File	1 KB	No	2 KB	65%	8/5/2025 4:29 PM
zap-report	JSON Source File	6 KB	No	44 KB	88%	8/5/2025 4:30 PM
zap-report	Markdown Source File	5 KB	No	36 KB	87%	8/5/2025 4:30 PM
zap-report	Chrome HTML Document	19 KB	No	121 KB	86%	8/5/2025 4:30 PM

Snippet from the ZAP scan report:

Site: http://54.152.164.138:30673

Generated on Tue, 5 Aug 2025 16:30:15

ZAP Version: 2.16.1

ZAP by Checkmarx

Summary of Alerts

Risk Level	Number of Alerts
High	0
Medium	2
Low	5
Informational	4
False Positives:	0

Summary of Sequences

For each step: result (Pass/Fail) - risk (of highest alert(s) for the step, if any).

Alerts

Name	Risk Level	Number of Instances
Content Security Policy (CSP) Header Not Set	Medium	11
Cross-Domain Misconfiguration	Medium	11
Cross-Domain JavaScript Source File Inclusion	Low	10
Dangerous JS Functions	Low	2
Deprecated Feature Policy Header Set	Low	11
Insufficient Site Isolation Against Spectre Vulnerability	Low	10
Timestamp Disclosure - Unix	Low	16
Information Disclosure - Suspicious Comments	Informational	2

References

Amazon Web Services (2025). *What Is Amazon EC2? - Amazon Elastic Compute Cloud*. [online] Amazon.com. Available at: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html.

Docker Documentation. (2024). *Docker Build Cloud*. [online] Available at: https://docs.docker.com/build-cloud/.

Openpolicyagent.org. (2025). *Using OPA in CI/CD Pipelines | Open Policy Agent*. [online] Available at: https://www.openpolicyagent.org/docs/cicd.

Openpolicyagent.org. (2025). *Policy Testing | Open Policy Agent*. [online] Available at: https://www.openpolicyagent.org/docs/policy-testing [Accessed 11 Aug. 2025].

Openpolicyagent.org. (2025). *Tutorial: Ingress Validation | Open Policy Agent*. [online] Available at: https://www.openpolicyagent.org/docs/kubernetes/tutorial [Accessed 11 Aug. 2025].

Istio. (2025). *Application Requirements*. [online] Available at: https://istio.io/latest/docs/ops/deployment/application-requirements/ [Accessed 11 Aug. 2025].

Zaproxy.org. (2025). *ZAP – ZAP vs OWASP Juice Shop*. [online] Available at: https://www.zaproxy.org/docs/scans/juiceshop/ [Accessed 11 Aug. 2025].

GitHub Docs. (2025). *Continuous integration - GitHub Docs*. [online] Available at: https://docs.github.com/en/actions/get-started/continuous-integration [Accessed 11 Aug. 2025].

Sonarsource.com. (2025). *Python test coverage | SonarQube Cloud Documentation*. [online] Available at: https://docs.sonarsource.com/sonarqube-cloud/enriching/test-coverage/python-test-coverage/ [Accessed 11 Aug. 2025].