Configuration Manual

MSc Research Project
Cloud Computing

Prajwal Kagganti Nataraja
Student ID: 23336251

School of Computing
National College of Ireland

Supervisor: Sai Gunaranjan Emani

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Prajwal Kagganti Nataraja
Student ID:	23336251
Programme:	MSc Cloud Computing **Year:** 2024-25
Module:	Research Project
Lecturer:	Sai Emani
Submission Due Date:	11-08-25
Project Title:	A Security-Centric Analysis of Declarative & Imperative Deployment Approaches in Kubernetes-Based Application Environments
Word Count:1610	**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: Prajwal K N

Date: 11-08-25

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):	

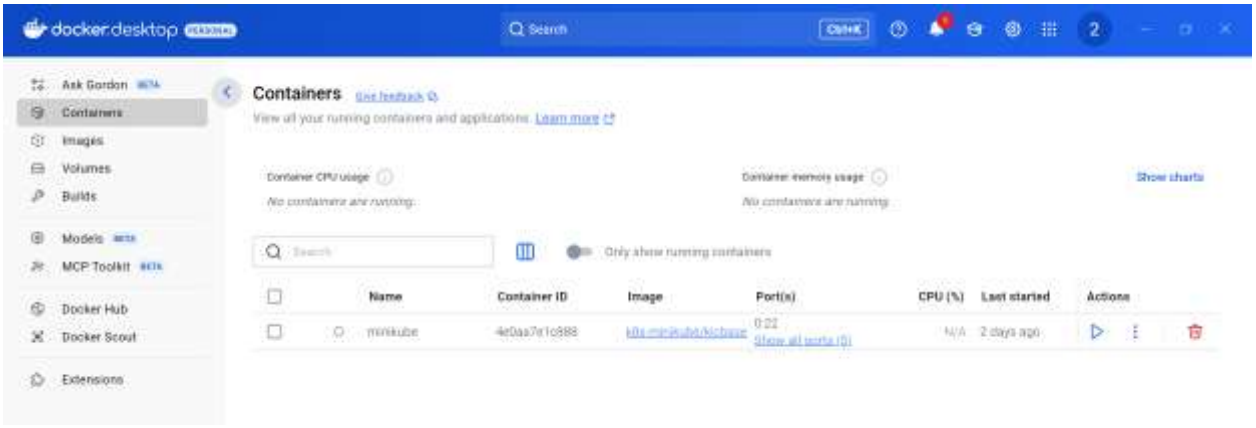
Configuration Manual

Prajwal Kagganti
23336251

Section 1:- Setting up the Environment

1.1.1 Step 1: Install Docker Desktop

1. Download Docker Desktop from: https://www.docker.com/products/docker-desktop.
2. Run the installer and proceed through the setup wizard.
3. Ensure “Enable WSL 2 feature” is selected during installation.
4. Restart the machine if prompted.
5. Launch Docker Desktop.
6. Ensure Docker is running (check “Engine running” on dashboard).



1.1.2 Step 2: Create Dockerfile for Flask App

1. Open PowerShell and navigate to your working directory:

```
cd D:\\Kube
```

2. Create a file named `Dockerfile` with the following content:

```
FROM python:3.13-slim
WORKDIR /app
COPY . /app
RUN pip install --no-cache-dir flask
EXPOSE 5000
CMD ["python", "app.py"]
```

1.1.3 Step 3: Create app.py for Flask App

1. In `D:\Kube`, create a file `app.py`:

```
from flask import Flask
app = Flask(__name__)

@app.route('/health')
```

```python
def health():
    return {"status": "healthy"}, 200

@app.route('/api/data')
def data():
    return {"message": "Sample data"}, 200

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

1.1.4 Step 4: Build and Run Flask Container

1. Open PowerShell and navigate to `D:\Kube`.

   ```
   cd D:\\Kube
   ```

2. Build Docker image:

   ```
   docker build -t flask-local .
   ```

3. Run the container:

   ```
   docker run -p 5000:5000 flask-local
   ```

4. Verify in browser: http://localhost:5000/health

1.1.5 Step 5: Install Minikube

1. Visit: https://minikube.sigs.k8s.io/docs/start/

2. Install using Chocolatey:

   ```
   choco install minikube
   ```

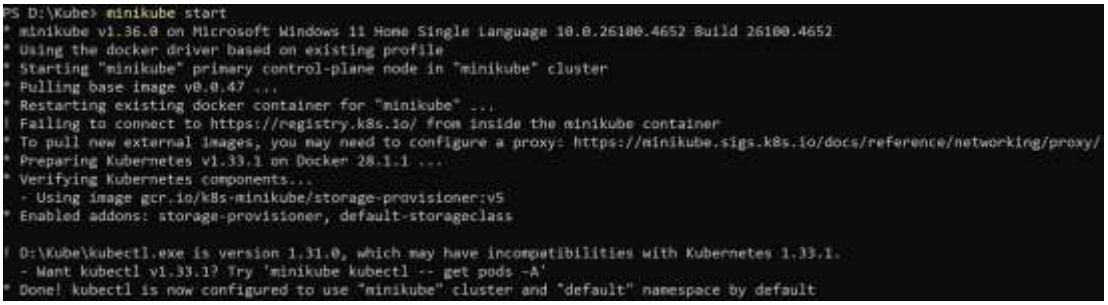
3. Verify:

   ```
   minikube version
   ```

```
PS D:\Kube> minikube version
minikube version: v1.36.0
commit: f8f52f5de11fc6ad8244afac475e1d0f96841df1-dirty
```

1.1.6 Step 6: Start Minikube

```
minikube start --driver=docker --cpus=2 --memory=4096mb
minikube status
```



```
PS D:\Kube> minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
```

Minikube start command and status check command

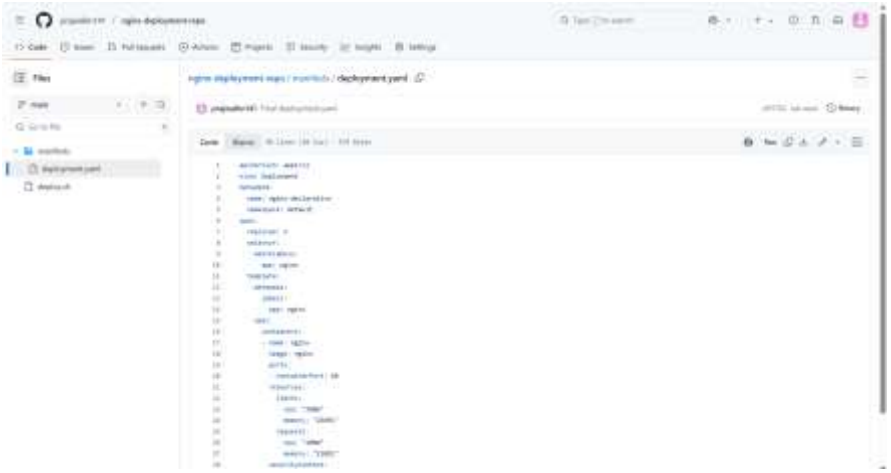
Section 2:- Creating Deployments

1.1.7 Step 7: Create deployment.yaml for Declarative Approach

1. In `D:\\Kube`, create `deployment.yaml`:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-declarative
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
        resources:
          limits:
            cpu: "200m"
            memory: "256Mi"
          requests:
            cpu: "100m"
            memory: "128Mi"
        securityContext:
          runAsNonRoot: true
          privileged: false
```

Same file is pushed to git with the folder location as /manifests/



1.1.8 Step 8: Apply Declarative Deployment

```
kubectl apply -f D:\Kube\deployment.yaml
kubectl get deployment nginx-declarative -n default
```

[Leave blank space for Screenshot 12: kubectl Apply Output]
[Leave blank space for Screenshot 13: kubectl Get Deployment Output]

1.1.9 Step 9: Install Argo CD for Declarative GitOps

```
helm repo add argo https://argoproj.github.io/argo-helm
helm repo update
helm install argo argo/argo-cd --namespace argo-cd --create-namespace
kubectl get pods -n argo-cd
```

Argo CD installation

```
PS D:\Kube> kubectl get pods -n default -l app=nginx
NAME                                  READY   STATUS    RESTARTS   AGE
nginx-declarative-96b9d695-29zhp      1/1     Running   0          3m48s
nginx-declarative-96b9d695-cn5w4      1/1     Running   0          3m47s
nginx-declarative-96b9d695-hm7qc      1/1     Running   0          18s
```

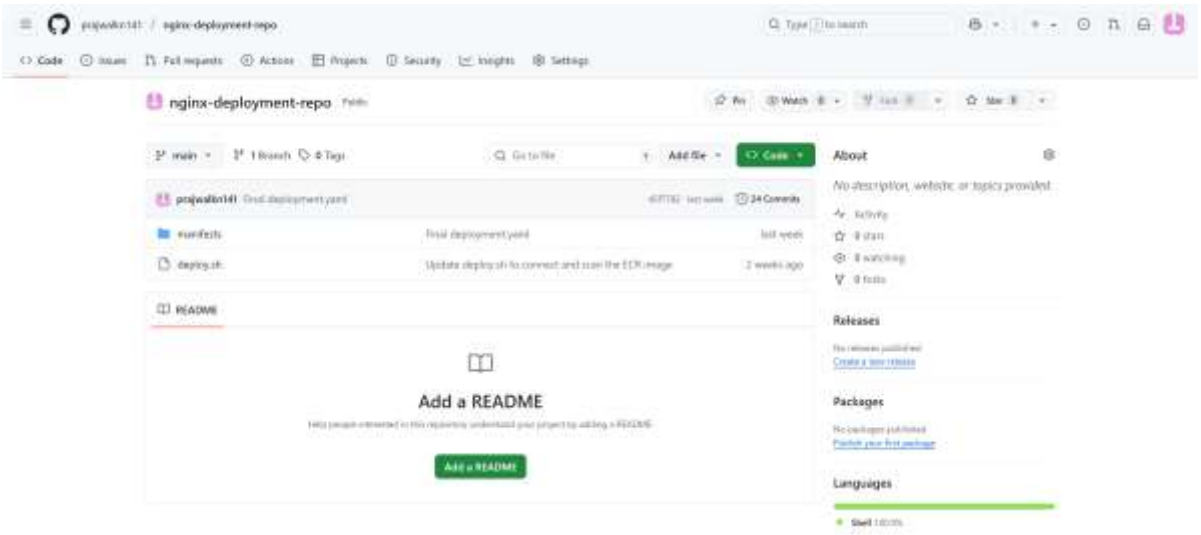
The pods status check for declarative

1.1.10 Step 10: Set Up Git Repo for GitOps

1. Create a GitHub repo: `nginx-deployment-repo`.

2. Initialize and push manifest:

```
git init
git add manifests/deployment.yaml
git commit -m "Add declarative deployment YAML"
git remote add origin https://github.com/prajwalkn141/nginx-deployment-repo.git
git push -u origin main
```



1.1.11 Step 11: Install Jenkins for Imperative Approach

```
helm repo add jenkinsci https://charts.jenkins.io
helm repo update
helm install my-jenkins jenkinsci/jenkins --namespace default --set
controller.admin.username="admin" --set controller.admin.password="Prajwal2025!" --set
persistence.enabled=false --set agent.enabled=true
kubectl get pods -n default
```

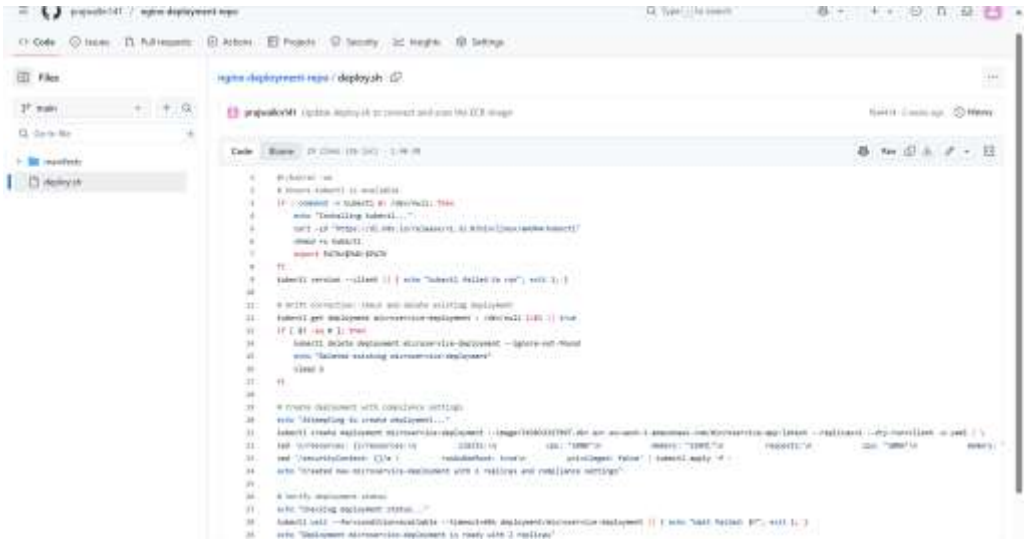
1.1.12 Step 12: Create deploy.sh for Imperative Deployment

Create `deploy.sh`:

```
#!/bin/sh -xe
if ! command -v kubectl &> /dev/null; then
  echo "Installing kubectl..."
  curl -LO "https://dl.k8s.io/release/v1.31.0/bin/linux/amd64/kubectl"
  chmod +x kubectl
  export PATH=$PWD:$PATH
fi
kubectl version --client || { echo "kubectl failed"; exit 1; }
kubectl get deployment nginx-imperative > /dev/null 2>&1 || true
if [ $? -eq 0 ]; then
  kubectl delete deployment nginx-imperative --ignore-not-found
  echo "Deleted..."
  sleep 5
fi
echo "Attempting to create..."
kubectl create deployment nginx-imperative --image=nginx:1.25-alpine --replicas=2 || {
echo "Create failed"; exit 1; }
```

```
echo "Created..."
echo "Checking..."
kubectl wait --for=condition=available --timeout=60s deployment/nginx-imperative || {
echo "Wait failed"; exit 1; }
echo "Deployment... is ready"
```

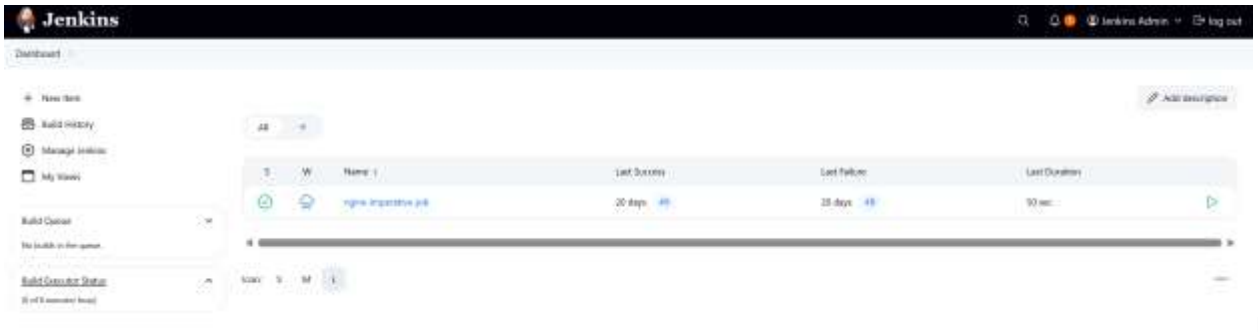
The deploy.sh file in Github



1.1.13 Step 13: Create Jenkins Job for Imperative Deployment

`kubectl port-forward svc/my-jenkins 8080:8080 -n default`

- Create job `nginx-imperative-job`
- Add Git repo, select “Execute shell” step with Trivy + `deploy.sh`



J

enkins job created (Freestyle job)

```
#!/bin/sh -xe
# Ensure kubectl is available
if ! command -v kubectl &> /dev/null; then
    echo "Installing kubectl..."
    curl -LO "https://dl.k8s.io/release/v1.31.0/bin/linux/amd64/kubectl"
    chmod +x kubectl
    export PATH=$PWD:$PATH
fi
kubectl version --client || { echo "kubectl failed to run"; exit 1; }

# Drift Correction: Check and delete existing deployment
kubectl get deployment microservice-deployment > /dev/null 2>&1 || true
if [ $? -eq 0 ]; then
    kubectl delete deployment microservice-deployment --ignore-not-found
    echo "Deleted existing microservice-deployment"
    sleep 5
fi

# Create deployment with compliance settings
echo "Attempting to create deployment..."
kubectl create deployment microservice-deployment --image=743833337997.dkr.ecr.eu-west-1.amazonaws.com/microservice-app:latest --replicas=2 --dry-run=client -o yaml | \
sed 's/resources: {}/resources:\n          limits:\n            cpu: "200m"\n            memory: "256Mi"\n          requests:\n            cpu: "100m"\n            memory:
sed '/securityContext: {}/a \          runAsNonRoot: true\n          privileged: false' | kubectl apply -f -
echo "Created new microservice-deployment with 2 replicas and compliance settings"

# Verify deployment status
echo "Checking deployment status..."
kubectl wait --for=condition=available --timeout=60s deployment/microservice-deployment || { echo "Wait failed: $?"; exit 1; }
echo "Deployment microservice-deployment is ready with 2 replicas"
```

e

xecute shell section of the Jenkins job

The Build successful in the Jenkins

1.1.14 Step 14: Install Kubescape

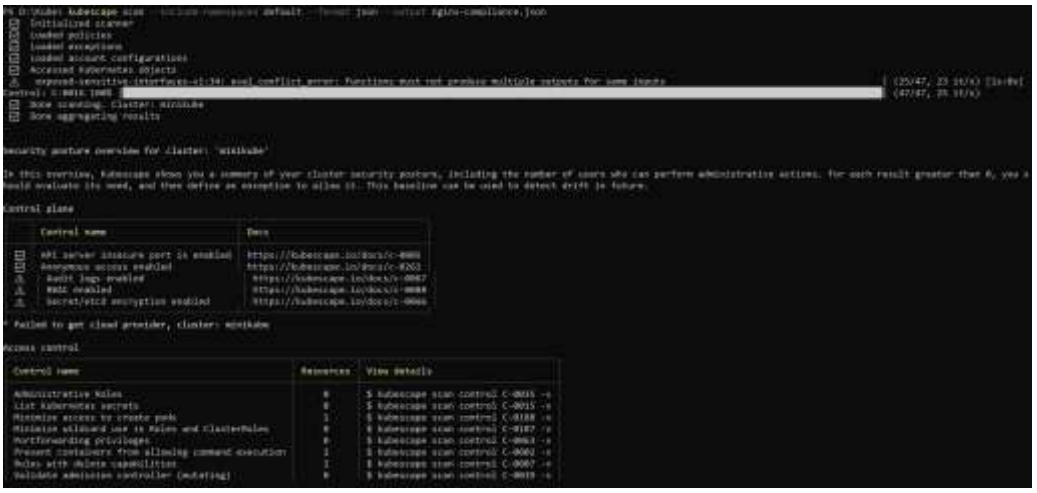
```
curl -s https://raw.githubusercontent.com/kubescape/kubescape/master/install.sh | /bin/bash
kubescape version
```

```
PS D:\Kube> kubescape version
Your current version is: v3.0.37
```

Section 3:- Testing the env and metrics

1.1.15 Step 15: Scan Declarative Deployment with Kubescape

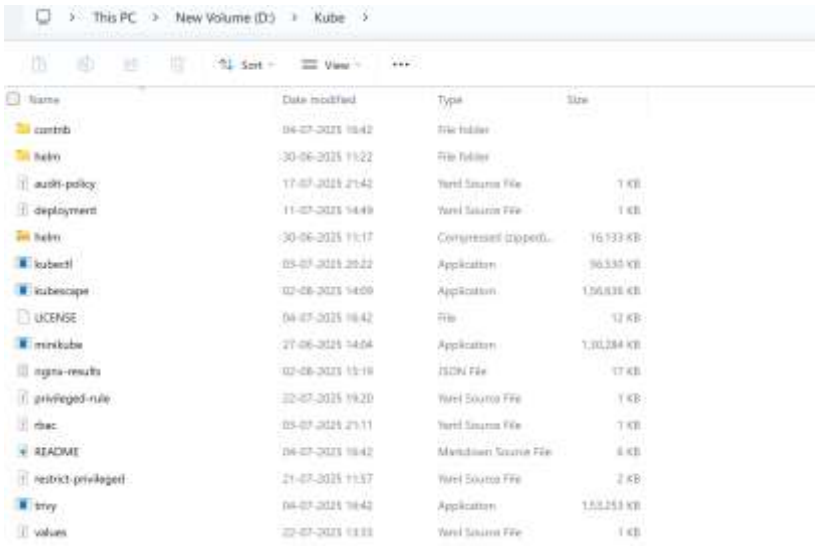
```
kubescape scan framework clusterscan D:\Kube\deployment.yaml --format json --output nginx-results.json
```



Kubescape scan results

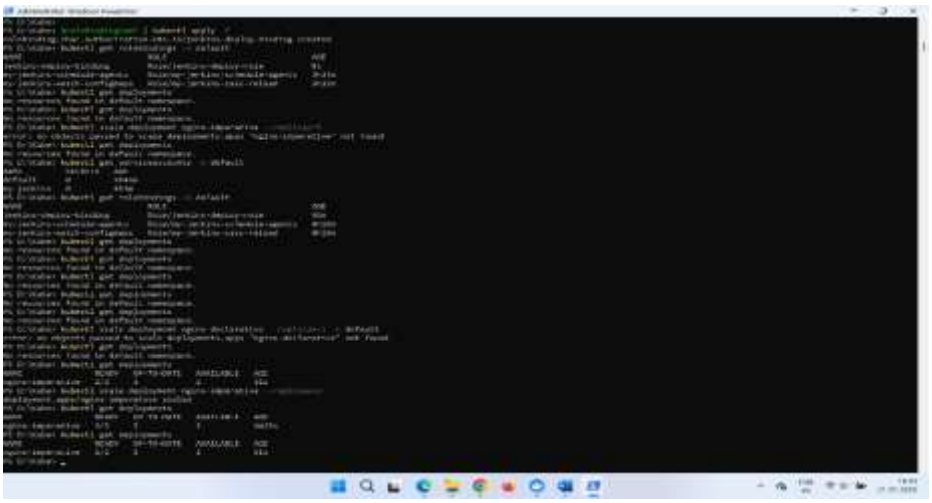
1.1.16 Step 16: Install Trivy

```
curl -sfL https://raw.githubusercontent.com/aquasecurity/trivy/main/contrib/install.sh | sh -x - --no-interactive
trivy --version
```



1.1.17 Step 17: Scan Imperative Image with Trivy

```
trivy image --exit-code 1 --severity CRITICAL,HIGH,MEDIUM --format table --output scan_imperative.txt nginx:1.25-alpine
```

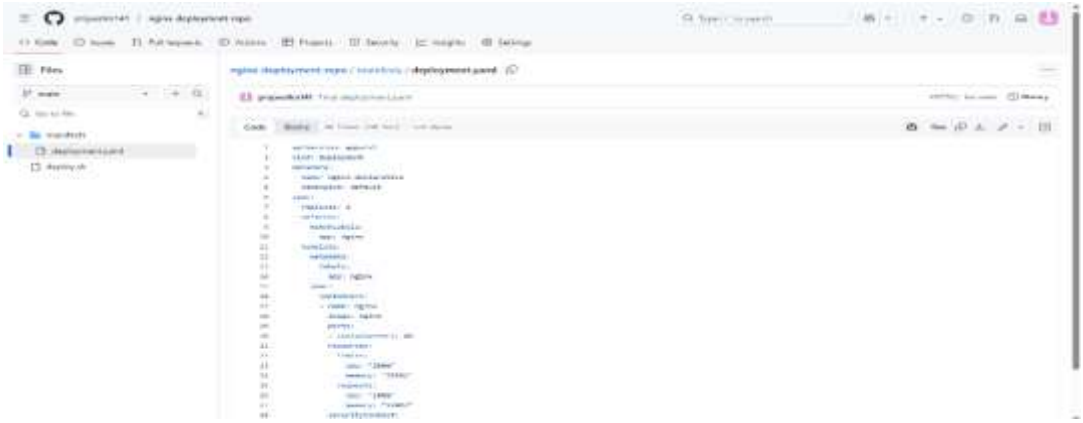


Trivy scan results for the imperative deployment

1.1.18 Step 18: Scan Declarative Image with Trivy

```
trivy image --severity CRITICAL,HIGH nginx:latest
```

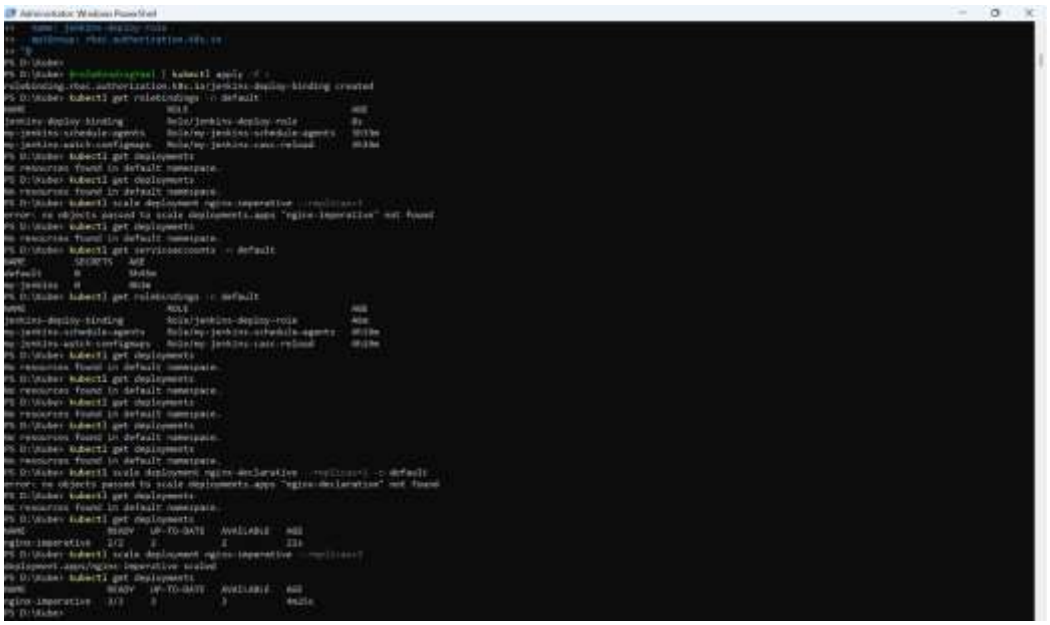
Trivy scan results for the declarative deployment



Deployment.yaml file which is used in declarative deployment

1.1.19 Step 19: Simulate and Test Drift Correction

```
kubectl scale deployment nginx-declarative --replicas=3
kubectl get deployment nginx-declarative
kubectl apply -f D:\Kube\deployment.yaml
```



Imperative approach drift correction(Manual)

Drift detection in Argo CD(automatic sync with no manual intervention)

References

Ahuja, N. (2023). Microservice Security and CI/CD Pipelines: Securing Kubernetes API Exposure, *International Journal of DevSecOps* 9(1), 34–42.

Aquasecurity (n.d.). Tracee Official Documentation.

URL: https://aquasecurity.github.io/tracee/v0.6.4/

Babu, Y. (2016). Docker Container Cluster Deployment Across Different Networks. MSc thesis, National College of Ireland, Dublin.

Civo (2025, January 21). How to Mitigate Kubernetes Runtime Security Threats.

URL: https://www.civo.com/learn/how-to-mitigate-kubernetes-runtime-security-threats

FluxCD Documentation (n.d.). Automating Kubernetes with Flux. URL: https://fluxcd.io/docs/

Guduru, S. (2019). Automated Vulnerability Scanning & Runtime Protection for Docker/Kubernetes: Integrating Trivy, Falco, and OPA, *The Journal of Scientific and Engineering Research* 6(2), 216–220.

Hung, P. N. T. (2024). Leveraging eBPF for Enhanced Kubernetes Observability and Security. MSc research project, National College of Ireland, Dublin.

Komodor (n.d.). Kubernetes Configuration Drift: Causes, Detection, and Prevention.

URL: https://komodor.com/learn/kubernetes-configuration-drift-causes-detection-and-

Kubernetes Documentation (n.d.). Kubernetes Concepts.
URL: https://kubernetes.io/docs/concepts/

Limoncelli, T. A. (2018). *The Practice of Cloud System Administration: Designing and Operating Large Distributed Systems*. Pearson Education.

Palo Alto Networks (n.d.). Best Practices for Kubernetes Security and RBAC Implementation.

URL: https://www.paloaltonetworks.com/resources/kubernetes-security

Prakash, R. (2024). Benchmarking Container Orchestration: Docker Swarm vs EKS,

International Journal of Computer Networks and Applications 11(1), 15–26.

Raftopoulos, J. (2025). Case Study: RBAC Misconfigurations in Production Kubernetes Clusters, *World Scientific News* 203, 336–373.

Roshan, P. (2025). Back to Basics with GitOps: Configuration Management Revisited,

Open Infrastructure Journal 6(1), 12–18.

SentinelOne Research (2025). Runtime Security in Containerized Workloads. SentinelOne Whitepaper. URL: https://www.sentinelone.com/resources/

Shekhar, R. (2019). Enhancing Kubernetes Container Scheduling Using Ant Colony Optimization, *Procedia Computer Science* 156, 140–148.

Shrestha, D. and Ali, R. (2024). Security Assessment of GitOps vs Imperative Configuration Management in Kubernetes, *Journal of Cloud Computing* 12(2), 203–215.

Solanki, M. (2024). Automated Drift Detection and Remediation in Infrastructure as Code Deployments, *IEEE Access* 12, 10121–10130.

Thiyagarajan, S. (2019). Automated Disaster Recovery using Terraform and Ansible. In:

International Conference on Automation and Cloud Computing.

Tigera (n.d.). What Is Kubernetes Security Posture Management (KSPM)? URL: https://www.tigera.io/learn/guides/kubernetes-security/kspm/

6(2), 216–220.

Hung, P. N. T. (2024). Leveraging eBPF for Enhanced Kubernetes Observability and Security. MSc research project, National College of Ireland, Dublin.

Komodor (n.d.). Kubernetes Configuration Drift: Causes, Detection, and Prevention.

URL: https://komodor.com/learn/kubernetes-configuration-drift-causes-detection-and-

Kubernetes Documentation (n.d.). Kubernetes Concepts. URL: https://kubernetes.io/docs/concepts/

Limoncelli, T. A. (2018). *The Practice of Cloud System Administration: Designing and Operating Large Distributed Systems*. Pearson Education.

Palo Alto Networks (n.d.). Best Practices for Kubernetes Security and RBAC Imple- mentation.

URL: https://www.paloaltonetworks.com/resources/kubernetes-security

Prakash, R. (2024). Benchmarking Container Orchestration: Docker Swarm vs EKS,

International Journal of Computer Networks and Applications 11(1), 15–26.

Raftopoulos, J. (2025). Case Study: RBAC Misconfigurations in Production Kubernetes Clusters, *World Scientific News* 203, 336–373.

Roshan, P. (2025). Back to Basics with GitOps: Configuration Management Revisited,

Open Infrastructure Journal 6(1), 12–18.

SentinelOne Research (2025). Runtime Security in Containerized Workloads. SentinelOne Whitepaper.

URL: https://www.sentinelone.com/resources/

Shekhar, R. (2019). Enhancing Kubernetes Container Scheduling Using Ant Colony Optimization, *Procedia Computer Science* 156, 140–148.

Shrestha, D. and Ali, R. (2024). Security Assessment of GitOps vs Imperative Configur- ation Management in Kubernetes, *Journal of Cloud Computing* 12(2), 203–215.

Solanki, M. (2024). Automated Drift Detection and Remediation in Infrastructure as Code Deployments, *IEEE Access* 12, 10121–10130.

Thiyagarajan, S. (2019). Automated Disaster Recovery using Terraform and Ansible. In:

International Conference on Automation and Cloud Computing.

Tigera (n.d.). What Is Kubernetes Security Posture Management (KSPM)? URL: https://www.tigera.io/learn/guides/kubernetes-security/kspm/