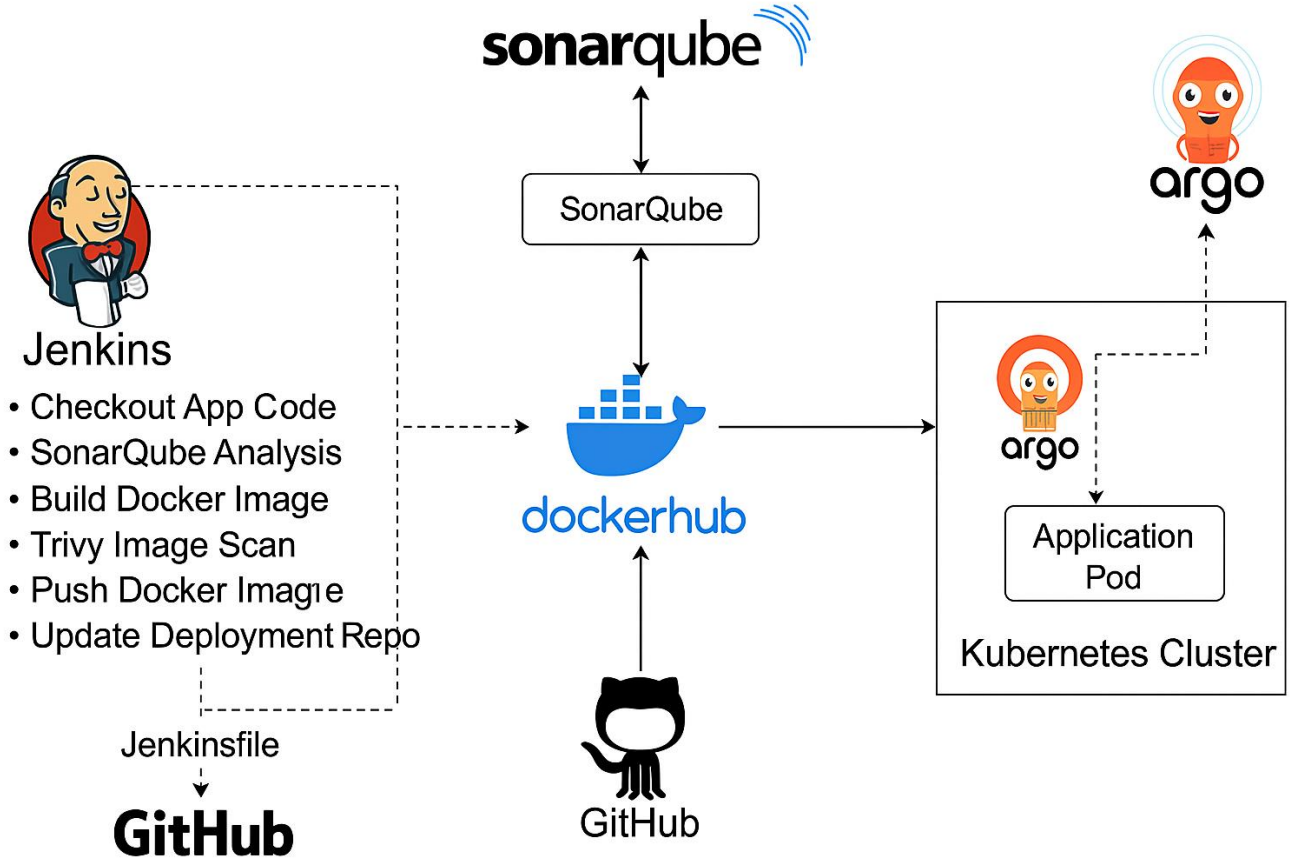
Task Report:

Architecture:



The goal was to make sure that as soon as code is pushed to github, it automatically goes through code quality checks, that is by using sonarqube, security scans using Trivy, builds a docker image, pushes it to dockerhub, updates a deployment repository which has been separately created, and finally gets deployed into a Kubernetes cluster using Argocd. Everything is automated from pushing code to deploying a live app.

Steps:

1. Created a VM on google cloud and then ssh into vm to set up the environment for example install Jenkins, kubectl, trivy, docker etc. Credentials for github, dockerhub and sonarqube token added in the Jenkins. Docker installed and added Jenkins user to docker group using

```
sudo usermod -aG docker jenkins
```

2. Created Kubernetes cluster (GKE) and connected it with the VM, for that I had to install gcloud and authenticated my gcp account. (service account created and attached with vm first with roles like Kubernetes engine permissions, compute engine permissions.)
   Command used to connect to GKE cluster is

```
gcloud container clusters get-credentials
```

   Verified connection using kubectl get nodes
3. Argocd installed on Kubernetes by using commands given on the official Argocd website and exposed using loadbalancer to access its UI.
4. After preparing everything, application code and deployment repositories created.
   Webapp repo has flask code, I have added Jenkins file, edited docker file, edited requirements.txt file, edited app.py
   Webapp-deploy repo created which has all the manifest files like deployment.yaml, secrets.yaml, service.yaml.
5. Jenkinsfile written for CI pipeline

It includes the stages like:

- Checkout app code from github repo (webapp)
- Sonarqube code analysis
- Buid docker image
- Trivy image scan
- Push the image to dockerhub
- Clone webapp-deploy repo to update the image tag, then commit and push

6. Sonarqube is installed and configured inside docker container by using this command:

```
docker run -d --name sonarqube -p 9000:9000 sonarqube
```

Exposed using vm's external ip: 9000

And token created for putting inside the Jenkins credentials as sonarqube-token

7. For trivy which scans docker image for vulnerabilities, in the Jenkins file

```
stage('Trivy Image Scan') {
steps {
  sh '''
```

```
        mkdir -p ./bin
            curl -sfL
 https://raw.githubusercontent.com/aquasecurity/trivy/main/contrib/install.sh |
 sh -s -- -b ./bin
            ./bin/trivy image ${IMAGE_NAME}:${TAG}
             '''
```

8. Created image after scan and then pushed to dockerhub..

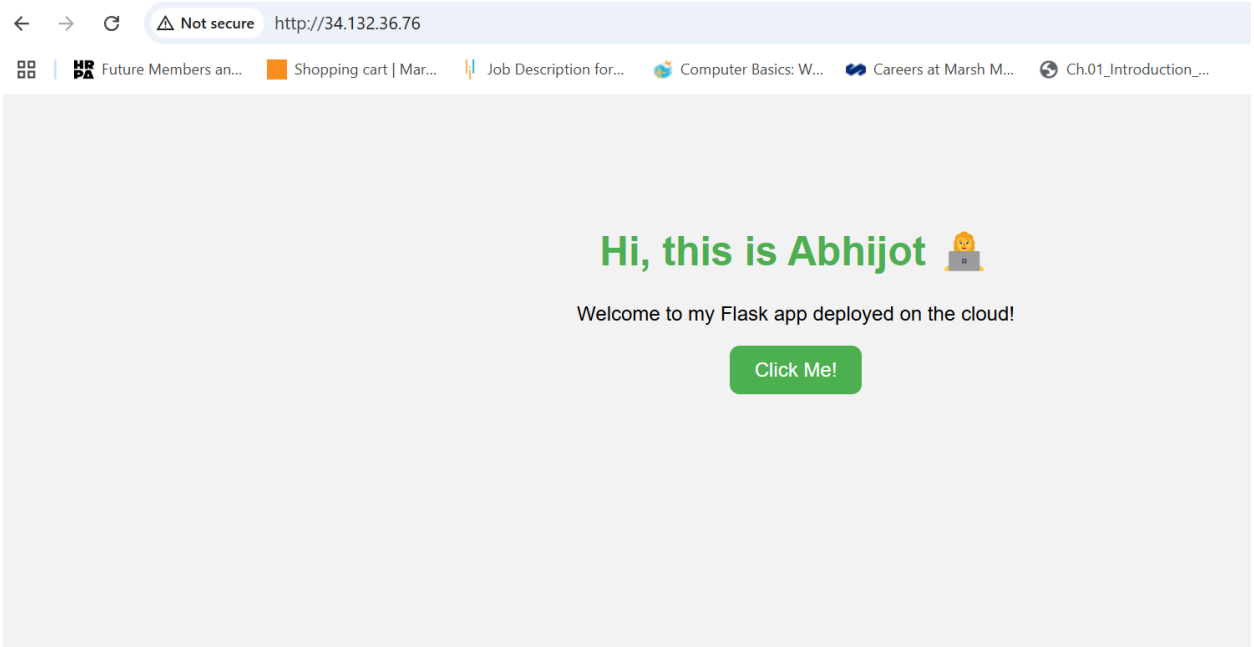
For this I have created a repository on the dockerhub and then the pushed image has the new tag (similar to the build number)

https://hub.docker.com/repository/docker/abkaur95/webapp/general

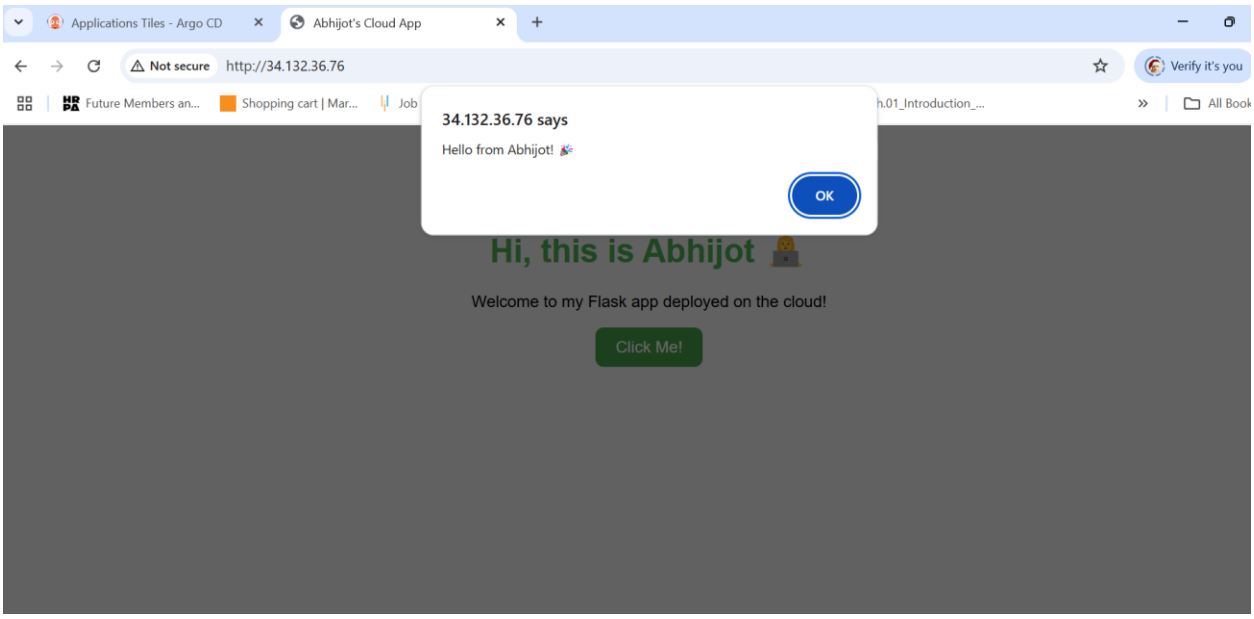
9. Along with image pushed to dockerhub, the deployment repo is also updated with new image version, for this in Jenkins stage, it will clone the repo first and replace the image tage inside deployment.yaml and then pushed the change.
10. This webapp-deploy repo is connected to argocd, basically argocd will watch this repo for changes, when a new version is pushed from Jenkins, argocd will sync the changes and deploys it to the Kubernetes cluster.
11. Application running with new image in the cluster, which is accessed through the external Ip from the cluster.

Outputs:

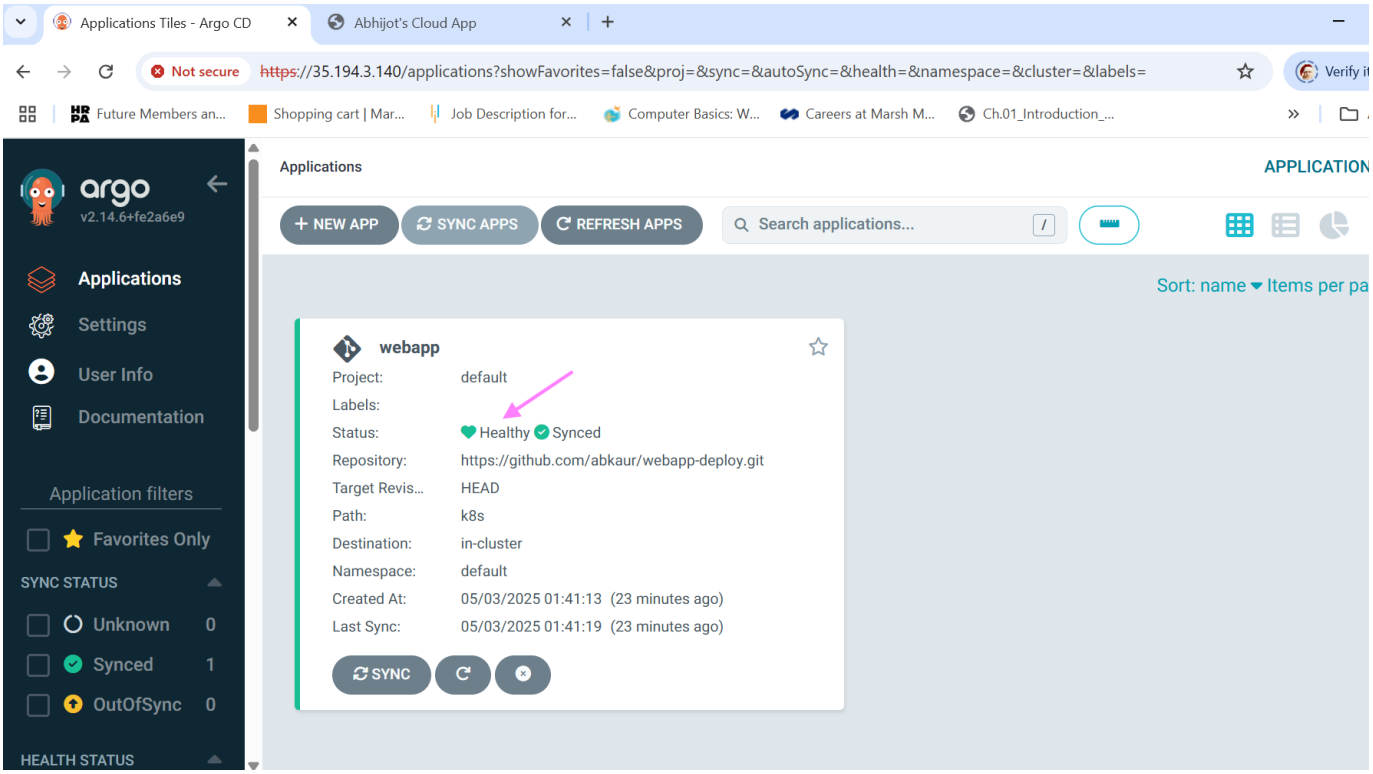
Final output- application running



By clicking on the button:



Argocd dashboard showing the healthy status



Showing services running in both namespaces: argocd and default

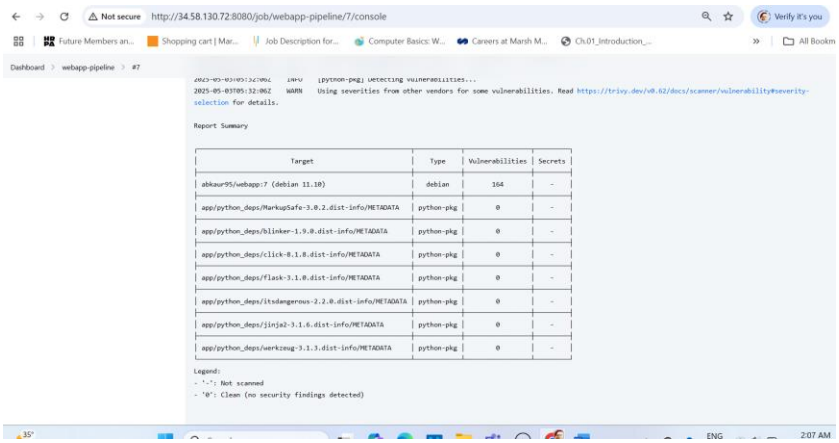
Loadbalancer in argocd namespace for argocd dashboard

Loadbalancer in default namespace for final application itself

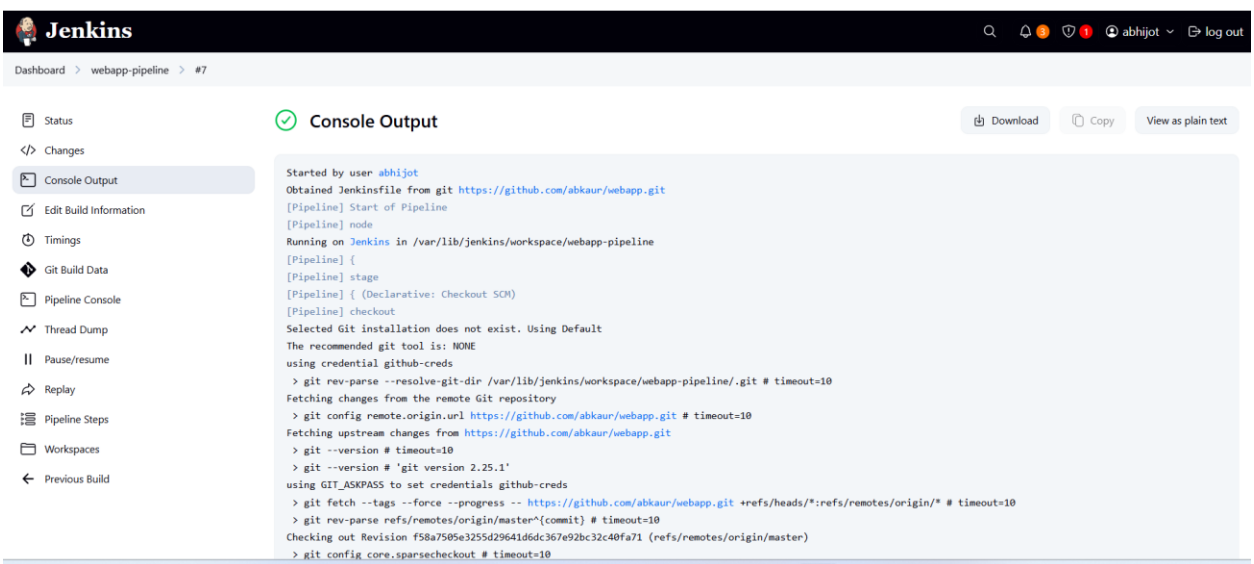
```
ka5828175@instance-jenkins:~$ kubectl get svc -n argocd
NAME                                      TYPE           CLUSTER-IP       EXTERNAL-IP     PORT(S)                      AGE
argocd-applicationset-controller          ClusterIP      34.118.226.122   <none>          7000/TCP,8080/TCP            94m
argocd-dex-server                         ClusterIP      34.118.227.141   <none>          5556/TCP,5557/TCP,5558/TCP   94m
argocd-metrics                            ClusterIP      34.118.225.216   <none>          8082/TCP                     94m
argocd-notifications-controller-metrics   ClusterIP      34.118.231.174   <none>          9001/TCP                     94m
argocd-redis                              ClusterIP      34.118.239.187   <none>          6379/TCP                     94m
argocd-repo-server                        ClusterIP      34.118.236.104   <none>          8081/TCP,8084/TCP            94m
argocd-server                             LoadBalancer   34.118.233.139   35.194.3.140    80:30087/TCP,443:32266/TCP   94m
argocd-server-metrics                     ClusterIP      34.118.227.120   <none>          8083/TCP                     94m
ka5828175@instance-jenkins:~$ kubectl get svc -n default
NAME             TYPE           CLUSTER-IP       EXTERNAL-IP    PORT(S)        AGE
kubernetes       ClusterIP      34.118.224.1     <none>         443/TCP        9h
webapp-service   LoadBalancer   34.118.227.233   34.132.36.76   80:31883/TCP   17m
ka5828175@instance-jenkins:~$
```

Showing the report in the jenkins output for trivy:

Trivy is used to scan the docker image for known vulnerabilities, it make sures the image we build does not contain security vulnerabilities before we push it to production.



Showing successful pipeline result in Jenkins



Build number #7

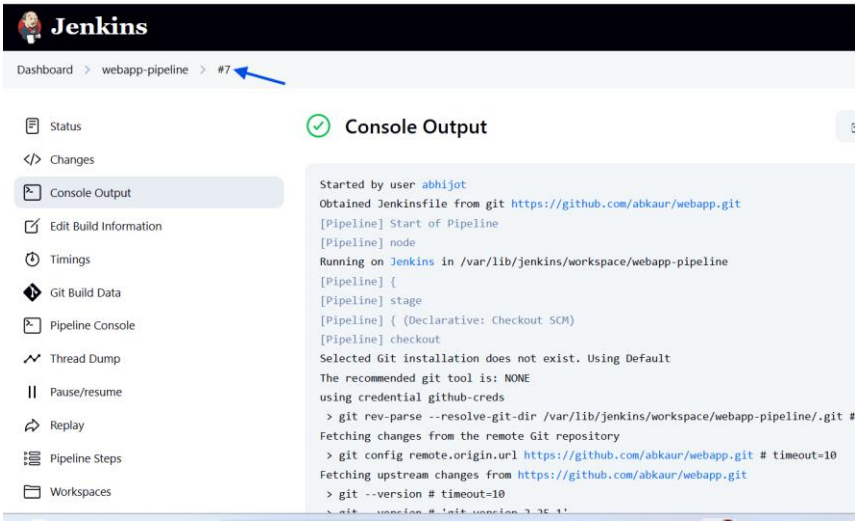
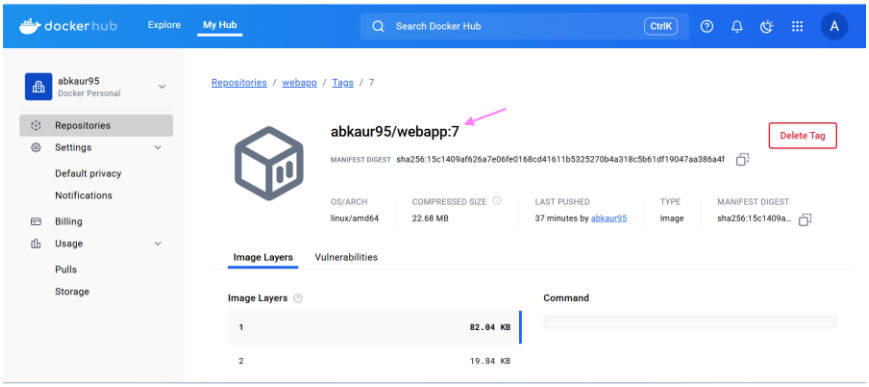


Image pushed with tag 7 in the dockerhub:



And similarly in the deployment.yaml file in the webapp-deploy repository, image tag got updated.

https://github.com/abkaur/webapp-deploy/blob/main/k8s/deployment.yaml

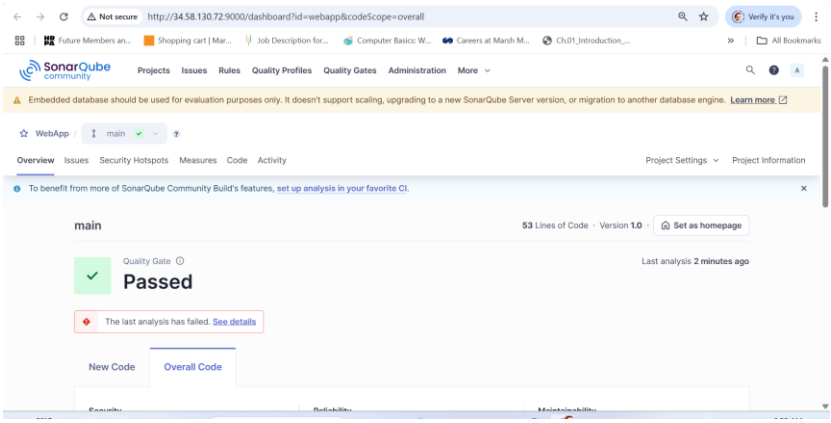
Future Members an... | Shopping cart | Mar... | Job Description for... | Computer Basics: W... | Careers at Marsh M...

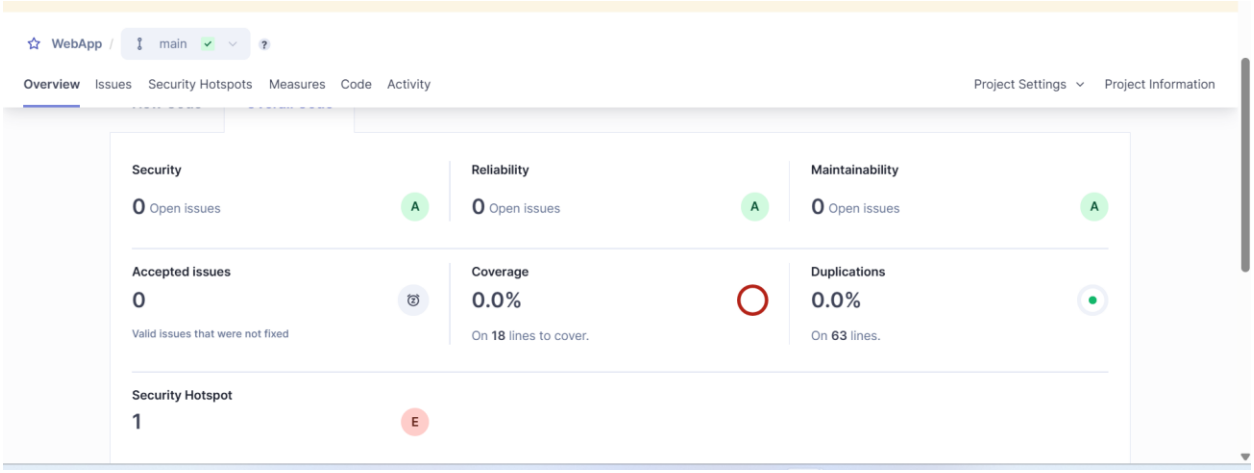
main webapp-deploy / k8s / deployment.yaml

Code Blame 27 lines (27 loc) · 541 Bytes

```yaml
spec:
  replicas: 2
  selector:
    matchLabels:
      app: webapp
  template:
    metadata:
      labels:
        app: webapp
    spec:
      containers:
        - name: webapp
          image: abkaur95/webapp:7
          ports:
            - containerPort: 5000
          env:
            - name: ADMIN_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: app-secret
                  key: ADMIN_PASSWORD
```

Sonarqube checks for bugs, code smells, security vulnerabilities and bad coding practices in the code. It gave a report on how clean and safe the code is, which helps improve the code quality early in the CI pipeline.




WebApp
main
Overview
Issues
Security Hotspots
Measures
Code
Activity
Project Settings
Project Information
Security
0 Open issues
A
Reliability
0 Open issues
A
Maintainability
0 Open issues
A
Accepted issues
0
Valid issues that were not fixed
Coverage
0.0%
On 18 lines to cover.
Duplications
0.0%
On 63 lines.
Security Hotspot
1
E