Configuration Manual

MSc Research Project
MSc Cloud Computing

Soham Yadav
Student ID: x23173394

School of Computing
National College of Ireland

Supervisor: Prof. Rashid Mijumbi

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Soham Yadav
Student ID:	x23173394
Programme:	MSc Cloud Computing
Year:	2024
Module:	MSc Research Project
Supervisor:	Prof. Rashid Mijumbi
Submission Due Date:	12/12/2024
Project Title:	Optimizing Kubernetes Security through automated Policy Enforcement in Multi-Cloud Environment
Word Count:	1573
Page Count:	17

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:	Soham Yadav
Date:	12th December 2024

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

Soham Yadav
x23173394

1 Introduction

This is a configuration manual that provides a detailed guidance for creating and configuring the Kubernetes Security Framework which includes Custom Security Scanning Agent, Adaptive Policy Enforcement, CI CD Pipeline and Cloud Services. This manual provides a step-by-step instructions and commands to configure required tools and technologies. It is created tailoring to a developer or a student seeking to recreate or utilize this framework as a base for the research. By following the steps, user can configure the framework and work over it.

2 Configurations

This section provides a table that provides information about the tools and technologies used in this research along with their versions.

Tools	**Version**
Java	17.0.11
Python	3.12.1
Maven	3.9.9
Docker Desktop	27.3.1, build ce12230
VS Code	1.95.3
Git	2.47.0.windows.2
Kubernetes / kubectl client	v1.30.5
Google Cloud SDK	502.0.0
helm	v3.16.2
Azure cli	2.67.0

3 Setup and Installations for Development Environment

3.1 Install and Setup Visual Studio Code

- Download VS Code by visiting "`https://code.visualstudio.com/download`"
- Click on the Download link based on the OS (Windows, MacOS, Linux) being used.
- Once the downloading is complete, run the installer.
- Select the location to store the IDE files.
- After successful Installation, open the IDE to verify if it's installed properly.

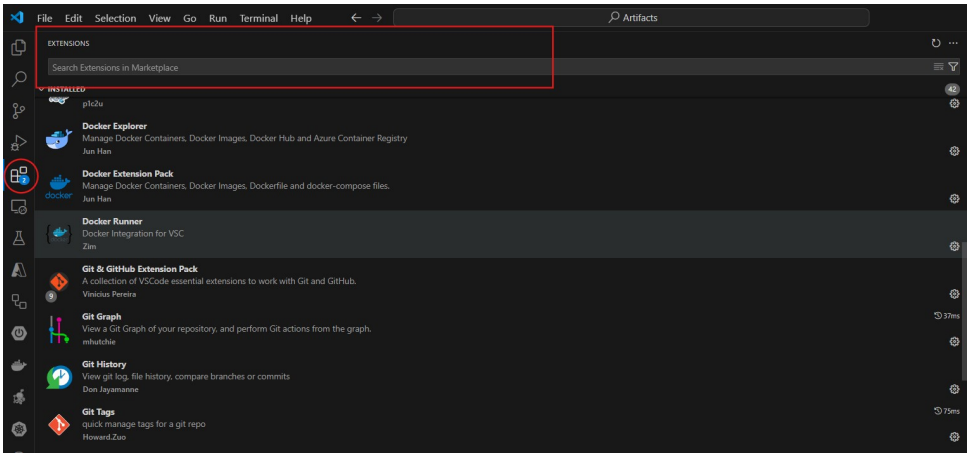

Figure 1: Extensions in VS Code

- Open the VS Code and navigate to the "Extensions" tab and install the following extensions, refer to image 1:
  1. Azure Accounts
  2. Azure Kubernetes Service
  3. Azure Services
  4. Docker
  5. Docker Extension Pack
  6. GitHub Actions
  7. GitLens
  8. Java
  9. Kubernetes
  10. Kubernetes Support
  11. Python
  12. Python Debugger

13. Spring Boot Extension Pack
14. Spring Boot Tools
15. WSL
16. YAML

- After the extensions are installed, restart VS Code to activate the extensions.

3.2 Install and Setup Git

- Download Git by visiting "`https://git-scm.com/downloads`"
- Click on the Download link based on the OS (Windows, MacOS, Linux) being used.
- Once the downloading is complete, run the installer.
- Open the "Command Prompt" or "Power Shell", refer to img2 and verify Git, by checking the Git version using the command **"git –version"**.

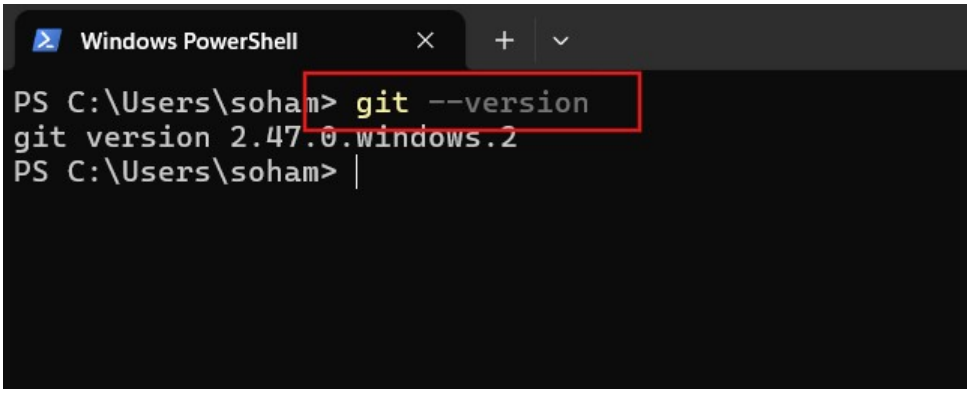

Figure 2: Git version

- If the Git installation is confirmed, configure Git:
  - **git config –global user.name "Your Name"**
  - **git config –global user.email "email@xyz.com"**
- verify the configuration by using the command **"git config –list"**

3.3 Other Setups (If not pre-existing)

1. Java Development Kit Setup:
   - Download the Java Development Kit by visiting the website "`https://www.oracle.com/ie/java/technologies/downloads/`"
   - Once the downloading process is completed, install Java by clicking on the installer.
   - Once the installation is in process, select the location to store JDK files/libraries.

- After installation, verify the installation by checking the Java version using the command **"java –version"** in "Command Prompt" or "Power Shell".

2. Maven Setup:

   - Download the Maven Kit by visiting the website "`https://maven.apache.org/download.cgi`"
   - Select "Binary Zip" and download it by clicking on the link in front of it "`https://dlcdn.apache.org/maven/maven-3/3.9.9/binaries/apache-maven-3.9.9-bin.zip`"
   - After the downloading process is complete, install it.
   - Post installation, verify the Maven version, by using the command **"mvn –version"** in "Command Prompt" or "Power Shell".

3.4 Install and Setup Docker

- Download the Docker desktop by visiting the website "`https://www.docker.com/products/docker-desktop/`" refer to image3

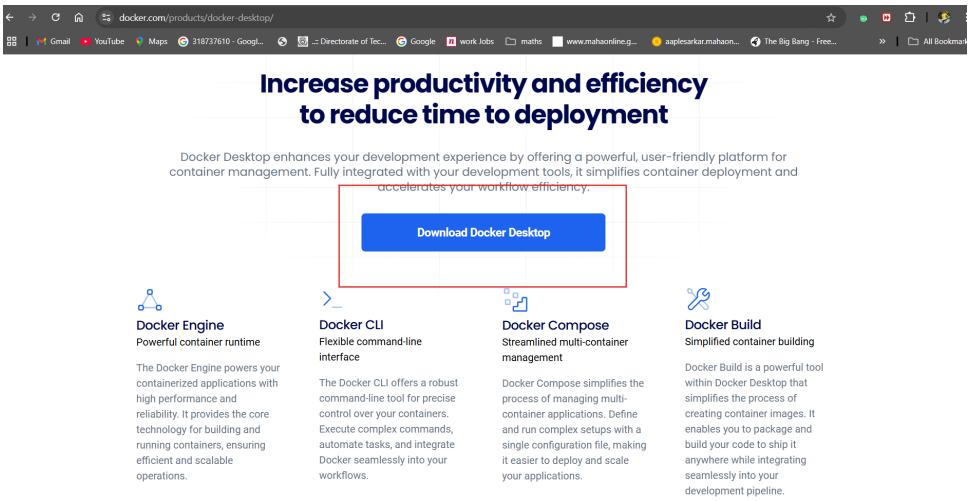

Figure 3: Docker Desktop download

- Click on the Download link based on the OS (Windows, MacOS, Linux) being used.
- Once the downloading process is completed, click on the installer to install and follow the prompts.
- Enable WSL 2 feature during installation
- Post successful installation, restart the system to apply the installation.
- Open the Docker desktop and wait until the start-up process is complete.
- Enable Kubernetes in docker, and wait until it starts completely.
- To validate the docker installation, open "Command Prompt" or "Power Shell" and enter the command **"docker –version"**.

- To verify docker functioning use the command **"docker run hello-world"** and it should provide a response "Hello from Docker!", refer to img4

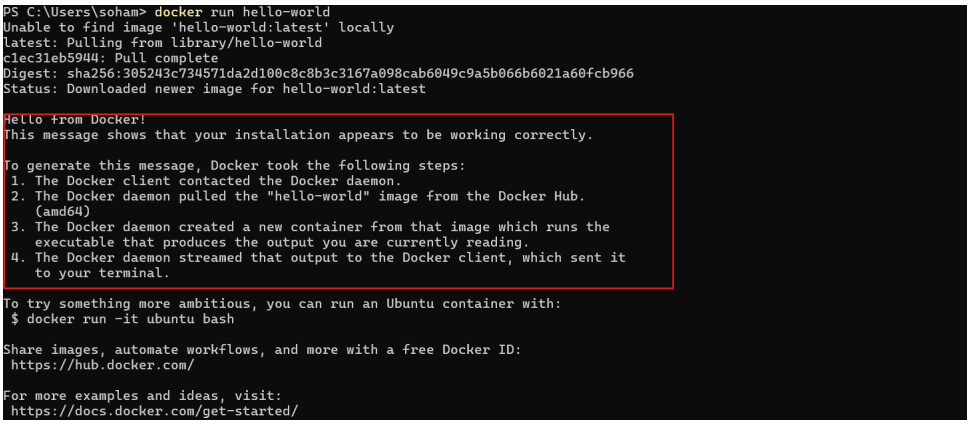

Figure 4: Hello From Docker

- Adjust the resources
  - Go to settings
  - click on Resources and then navigate to Advanced.
  - Set CPU and Memory limits to 2 CPUs and 2 GB respectively.
  - restart again to apply the settings.

3.5 Install and Setup Kubectl for Kubernetes Management

- open terminal or Command prompt or Power Shell.
- (prerequisite)If Chocolatey CLI is not installed use **"Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::SecurityProtocol = [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))"** for Power Shell, refer to image5.

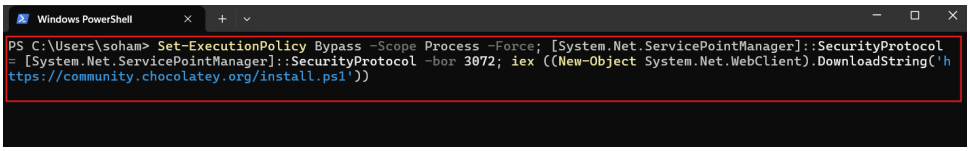

Figure 5: Install Kubectl

- Install Kubectl for Kubernetes using the command **"choco install kubernetes-cli"**.
- Verify the installation by checking the Kubectl version using **"kubectl version –client"**

3.6 Install and Setup Azure cloud

- (For this research I have used a free account of Kubernetes, that provides 200 euros as credits to use for a period of 1 month)
- To create an Azure account, visit the website "`https://azure.microsoft.com/en-us/pricing/purchase-options/azure-account`", refer to image6.

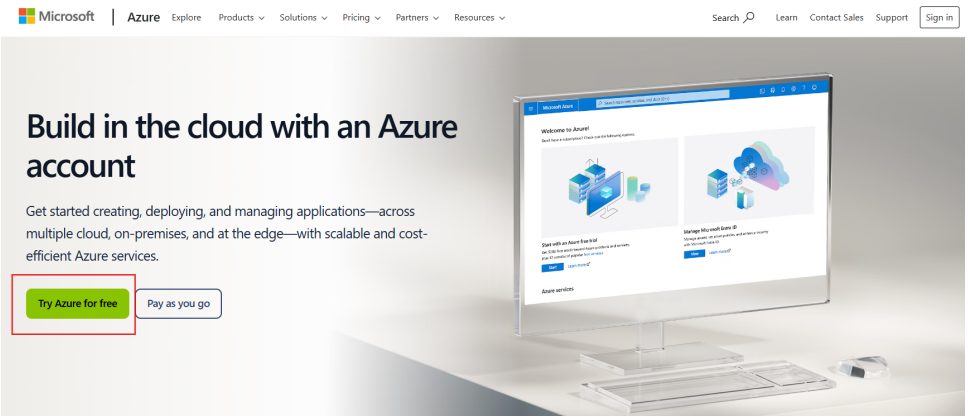

Figure 6: Azure free account

- Click on Sign in and provide all the required details.
- Provide Card details for payment, it will not ask for any payment as it provides 200 euros of credits initially.
- Once all the details are entered, you have setup your Azure account successfully. Refer to img 7

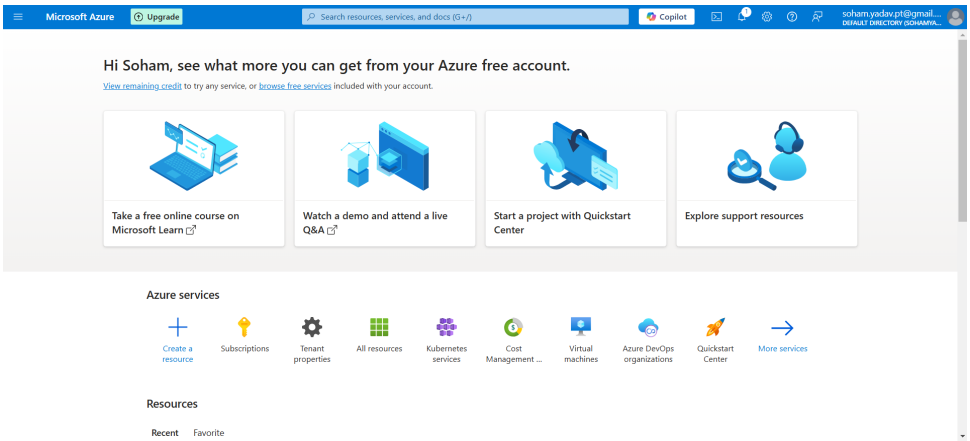

Figure 7: Azure account

4 Setup Project Framework

- Download or pull git repository from the "`https://github.com/Sohamy1999/kubernetes_security`" in your system, reefer to image8.

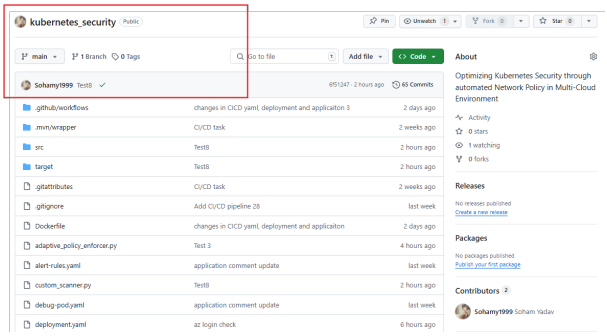

Figure 8: Git repo

- After downloading or pulling the repository in the VS Code at the desired workspace, use the command "cd security framework", to enter the Maven project folder.
- The Project folder structure is as follows, refer to img9:

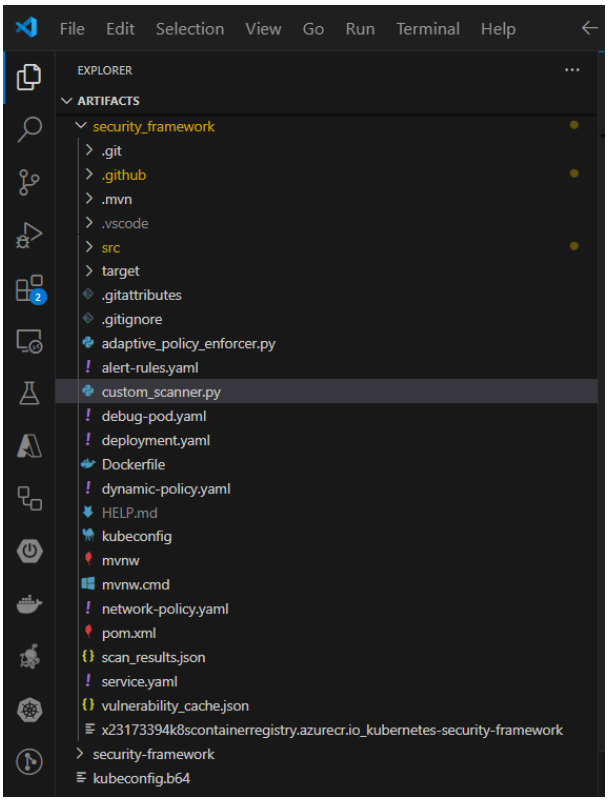

Figure 9: File Structure

  - pom.xml (File that stores all dependencies required for the project)

- github
  - ci cd pipeline.yaml
- .gitignore
- src
  - SimpleApplication.java
  - application.properties
- custom scanner.py (Custom Security Scanning Agent script)
- adaptive policy enforcer.py (Adaptive Policy Enforcement script)
- deployment.yaml (Kubernetes deployment configuration)
- service .yaml (Kubernetes service configuration)
- Dockerfile (Docker build image configuration)
- scan results.json (Vulnaneribilities found by custom scanner script gets stored here)
- dynamic policy.yaml (Policies generated by adaptive policy enforcer are stored here, which are used later for enforcement)

- After the project is completely imported, run "mvn clean install" to configure and update the project build.

5 Docker and Kubernetes configuration

5.1 Docker configuration

- GO to the VS Code, enter into the parent project folder (i.e. security framework)
- open terminal in VS Code at **"security framework"** level.
- (Dockerfile is already created and available in the project), to create docker image, run the command **"docker build -t < image-name > :< tag > "**,
  for e.g. (docker build -t kubernetes-security:latest)
- to verify if the image is created or not, enter the command "docker images" to get a list of images including the one created in the previous step.

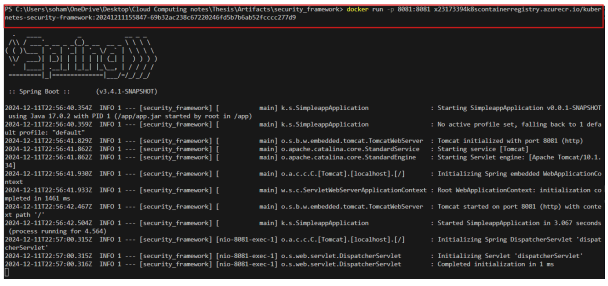

Figure 10: Docker Run

- Verify if the application is running properly by entering the command **"docker run -p 8080:8080 < image-name > :< tag > "**, refer to img10
- The application should be accessible over **"http://localhost:8080"**

5.2 Azure Kubernetes Configuration

- Firstly download Azure CLI by visiting the website "`https://learn.microsoft.com/en-us/cli/azure/install-azure-cli-windows?tabs=azure-cli`", and select the download link as per your system configuration.
- After the downloading process is complete, install Azure CLI using the installer.
- Open PowerShell and enter the command **"az –version"** to verify successful installation of Azure CLI, refer to img11.

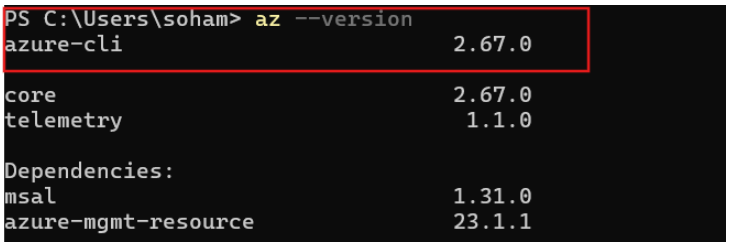

Figure 11: Azure version verify

- To login into Azure, enter the command **"az login"** in PowerShell, and a window will pop up asking for Azure login credentials, and it will log in successfully, refer to img 12.

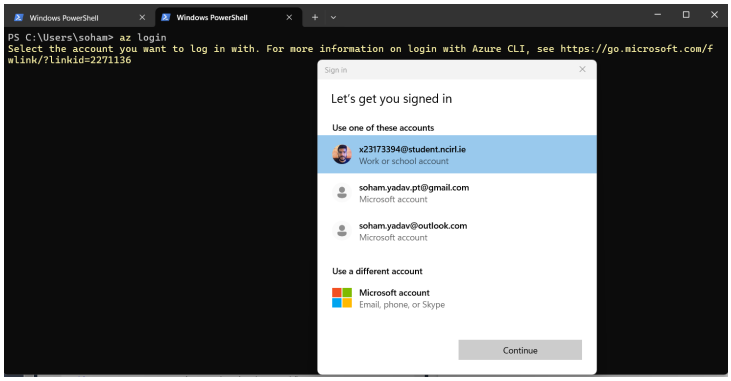

Figure 12: Azure login pop up window

- Navigate back to VS Code to the "security framework" level, and do **"az login"** again in the terminal and confirm login.
- Install Kubectl through Azure by using the command **"az aks install-cli"** and verify the installation using the command **"kubectl version –client"**.

- Create a resource group in Azure, by using the command **"az group create –name < resourcegroup-name > –location < location > "**, for e.g. (az group create –nameresourcegroup –location eastus)

- Create Azure Container Registry (ACR) by using the command **"az acr create –resource-group < resource-group-name > –name < registry-name > –sku Basic"**. sku is sufficient for a free version subscription.

- Verify ACR by using the command **"az acr list –resource-group < resource-group-name > –output table"** that generates a list of ACR.

- Enable admin access to ACR by using the command **"az acr update –name < registry-name > –admin-enabled true"**

- Retrieve the ACR credentials by using the command **"az acr credential show –name < registry-name > "**, It will provide username and password. Note those credentials for future use.

- Create Kubernetes Cluster by using the command **"az aks create –resource-group < resource-group-name > –name < cluster-name > –node-count 2 –enable-addons monitoring –generate-ssh-keys"**.

- Get the credentials for the created cluster by using the command **"az aks get-credentials –resource-group < resource-group-name > –name < cluster-name > "**

- verify if the connection to the cluster is created by using the command **"kubectl get pods"**

- Tag the above-created docker image to the ACR by using the command **"docker tag < image-name > :< tag > < registry-name > .azurecr.io/< image-name > :< tag > "**, for e.g. (docker tag myapp:latest mycontainerregistry.azurecr.io/myapp:latest).

- Login into ACR by using the command **"az acr login –name < registry-name > "**

- Push docker image to ACR by using the command **"docker push < registry-name > .azurecr.io/< image-name > :< tag > "**.

- Create Azure secrets by using command **" kubectl create secret docker-registry acr-secret‘ –docker-server=< registry-name > .azurecr.io ‘ –docker-username=< ACR-username > ‘ –docker-password=< ACR-password > "**

Possible Issues while configuring Kubernetes are as follows:

1. Azure subscription is not registered to use microsoft.insights namespace, used for monitoring.

    - Use the command **"az provider register –namespace Microsoft.Insights"**

- After the command, recreate the cluster using the command mentioned above.
- If the error persists, use this command to create the cluster **"az aks create –resource-group < registry-name > –name < cluster-name > –node-count 2 –generate-ssh-keys"**, it skips the add-ons insights.

2. Azure subscription is not registered to use Microsoft.ContainerService, used for managing AKS cluster.

   - Use the command **"az provider register –namespace Microsoft.ContainerService"**
   - After the command, recreate the cluster using the command mentioned above.
   - Also update the azure by using the command **"az update"**, to update all the configurations.

3. Azure resource provider issue.

   - Use the command **"az provider register –namespace Microsoft.Compute"**
   - Verify provider registration by using the command **"az provider show –namespace Microsoft.Compute –query "registrationState""**
   - After the command, request a Quota increase through the Azure dashboard.

5.3 Custom Security Scanning Agent and Adaptive Policy Enforcement

- Navigate to the "security framework" folder in VS Code.

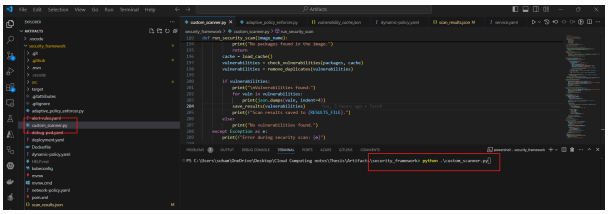

Figure 13: Initiate Custom Security Scanning Agent

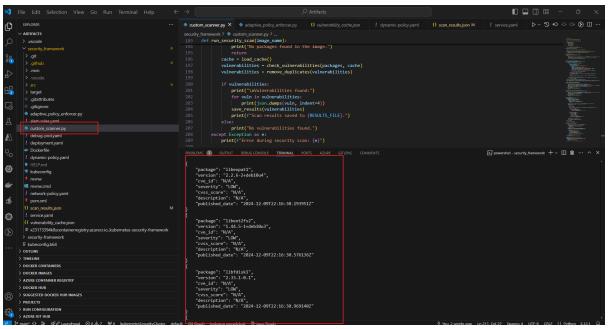

Figure 14: Vulnerabilities detected

- Run the scanner script by using the command **"python custom scanner.py"** and it should generate output with vulnerabilities if available in the image or with no vulnerabilities. refer to img 13 14

- If the vulnerabilities are present, it will be stored in "scan results.json" file. refer to img 15

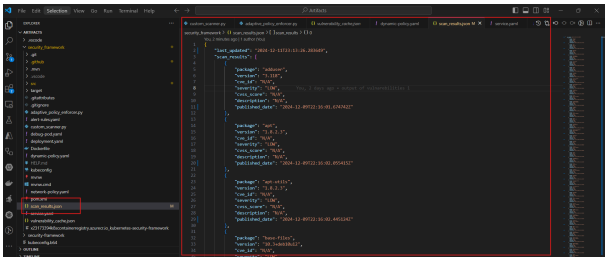

Figure 15: Scan Results Json

- Run policy enforcer by using the command **"python dynamic policy enforcer.py"**, and it will generate "dynamic policy.yaml" file with policies in it and will apply the policies. Ref to img 16 17

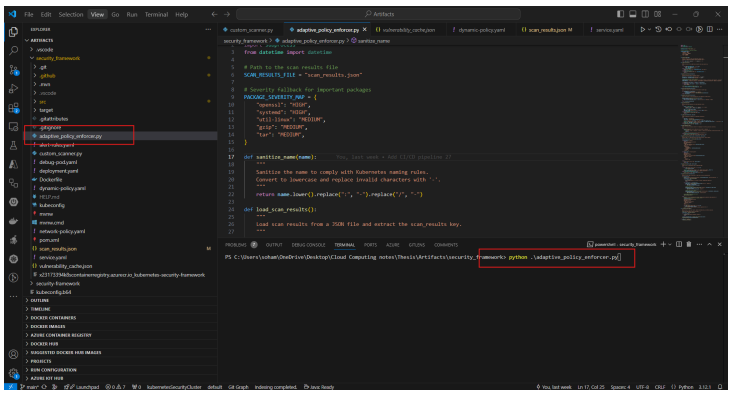

Figure 16: Run Adaptive policy Enforcement

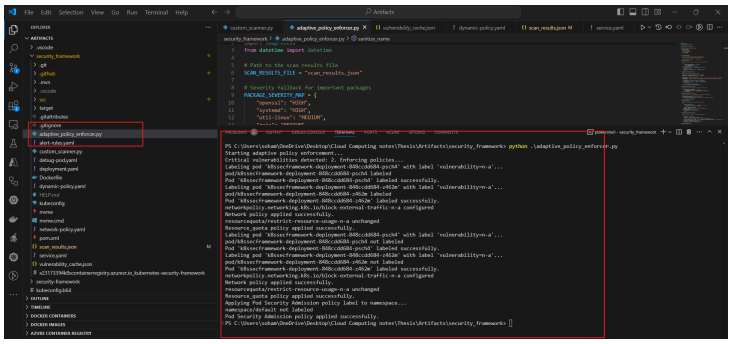

Figure 17: Applied Policies

5.4 Azure Kubernetes Deployment

- Go to "deployment.yaml" in the VS Code, inside the project folder and replave the image value with **"image: < your-acr-name > .azurecr.io/< image-name > :< tag > "**, for e.g. (image: < your-acr-name > .azurecr.io/kubernetes-security-framework:latest), refet to img18

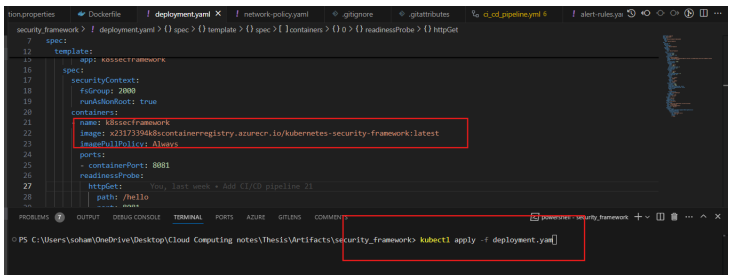

Figure 18: Deployment yaml

- Then use the command for deployment **"kubectl apply -f deployment.yaml"**
- After deployment apply service by using the command **"kubectl apply -f service.yaml"**

5.5 CI CD Pipeline Configuration

- Firstly Configure Git in the project folder by using commands:
  git init
  git add .
  git commit -m "comment"
  git remote add origin < repo url >
  git branch -m master main
  git push -u -f origin main
- Create Git secrets by visiting Git website and navigate to the repository
- Enter the settings and navigate to "secrets and variables" and select Actions, create secrets by the following variables and values:
  AZURE_PASSWORD = < ACR username >
  AZURE_USERNAME = < ACR Password >
  GH_PAT = < Git access code >
  KUBECONFIG = < Base 64 encoded kubeconfig >
- Navigate to "ci cd pipeline yaml" file under github/workflows folder in VS Code and change the registry name and image name wherever visible. refer to img19
- Then again commit new changes and push to git, and check the Git Actions for pipeline execution.

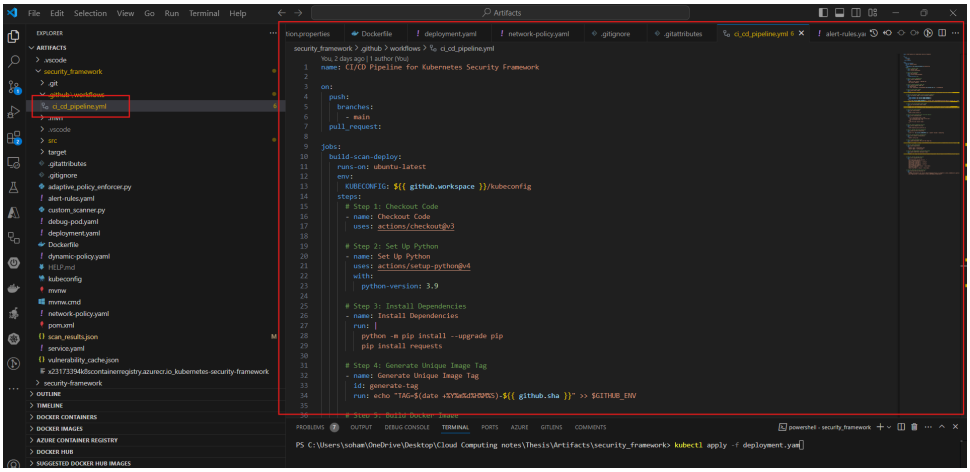

Figure 19: CI CD Pipeline YAML

5.6 Google cloud Configuration

- Visit the Google Cloud website "`https://cloud.google.com/?_gl=1*d245yx*_up*MQ..&gclid=CjwKCAiA6t-6BhA3EiwAltRFGGnk0E2vflVyUxHzZ-ozlX3oaZ40Tu-jqXOgDFHKMU GgQAvD_BwE&gclsrc=aw.ds&hl=en`" and create a free account by following the prompts.

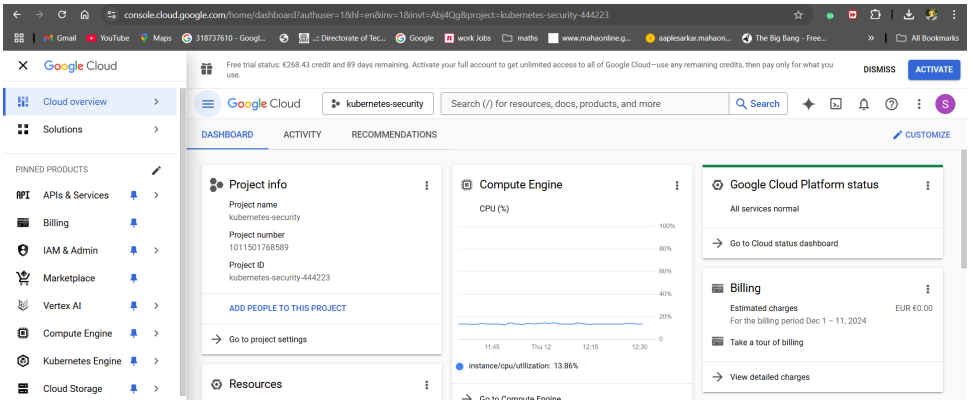

Figure 20: Google account

- After creating the account, refer to img20, enter the dashboard and click on "New Project"
- Enter the project name and create the project.
- Navigate to APIs and Service and go to the library and enable "Kubernetes Engine API", "Compute Engine API", and "Cloud Storage API".
- Download Google Cloud SDK from "`https://cloud.google.com/sdk/docs/install`"
- After downloading is completed, install the Google Cloud SDK using the installer.
- Log in the Google Cloud using the command **"gcloud init"**, and use the credentials used for creating the account.

- create Google Cloud cluster by using the command
  " gcloud container clusters create kubernetes-security-cluster '
  –num-nodes=2 '
  –region=us-central1 '
  –enable-ip-alias '
  –disk-size=50 "

- Verify nodes created in the cluster using the command **"kubectl get nodes"**

- Set up Google Cloud Registry by using the command
  "gcloud services enable containerregistry.googleapis.com"

- Configure Docker with gcloud using the command **"gcloud auth configure-docker"**

- Once the Docker is configured, tag Docker image to gcloud registry using the command **"docker tag < image-name > :< tag > gcr.io/< PROJECT_ID > /< image-name > :< tag > "**

- Push the image to gcloud by using the command
  "docker push gcr.io/< PROJECT_ID > /< image-name > :latest"

- Then follow the same process like make changes in deployment.yaml file and CI CD pipeline yaml and they apply deployment.

6 Monitoring

- Install Helm packages using the command **"curl https://raw.githubusercontent.com/helm/h /main/scripts/get-helm-3 bash"**

- create a monitoring namespace using the command **"kubectl create namespace monitoring"**

- Install Prometheus by using the command **"helm repo add prometheus-community https://prometheus-community.github.io/helm-charts"** followed by the command **"helm repo update".**

- Install Prometheus using helm by using the command **"helm install prometheus prometheus-community/prometheus –namespace monitoring"**

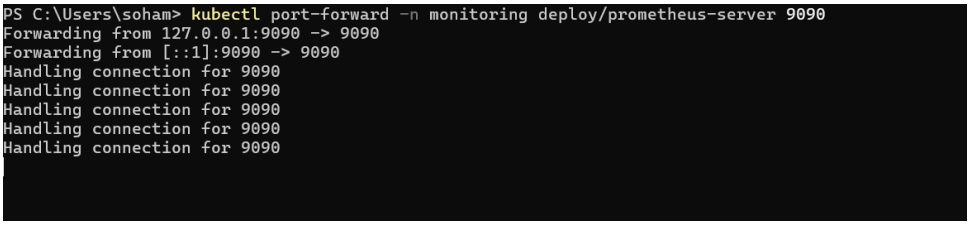

Figure 21: Initiate Prometheus

- Prometheus UI can be accessed by using the command **"kubectl port-forward -n monitoring deploy/prometheus-server 9090"** and then by the link **"http://localhost:9090/"**, refer to img 21 22.

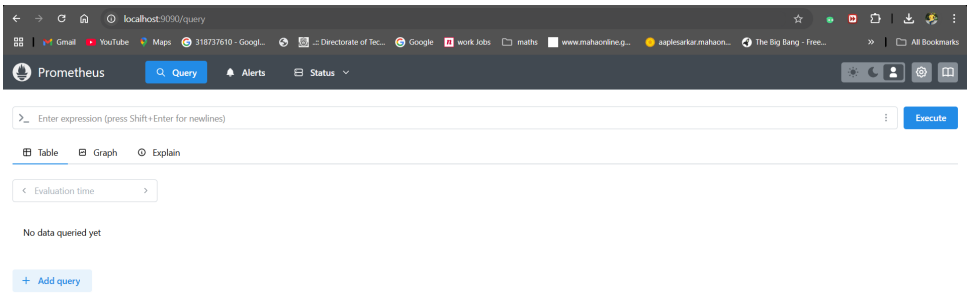

Figure 22: Prometheus Dashboard

- Configure Grafana by using the command **"helm repo add grafana https://grafana.github.io/helm-charts"** and then update the helm repo.
- Install Grafana using the command **"helm install grafana grafana/grafana –namespace monitoring"**
- Grafana UI can be accessed using the command **"kubectl port-forward -n monitoring deploy/grafana 3000"** and by using the link **"http://localhost:3000/"**, refer to img 23.

Figure 23: Initiate Grafana

- Login into the Grafana dashboard and configure Prometheus url to get data visuals on the Grafana dashboard. Refer to img 24

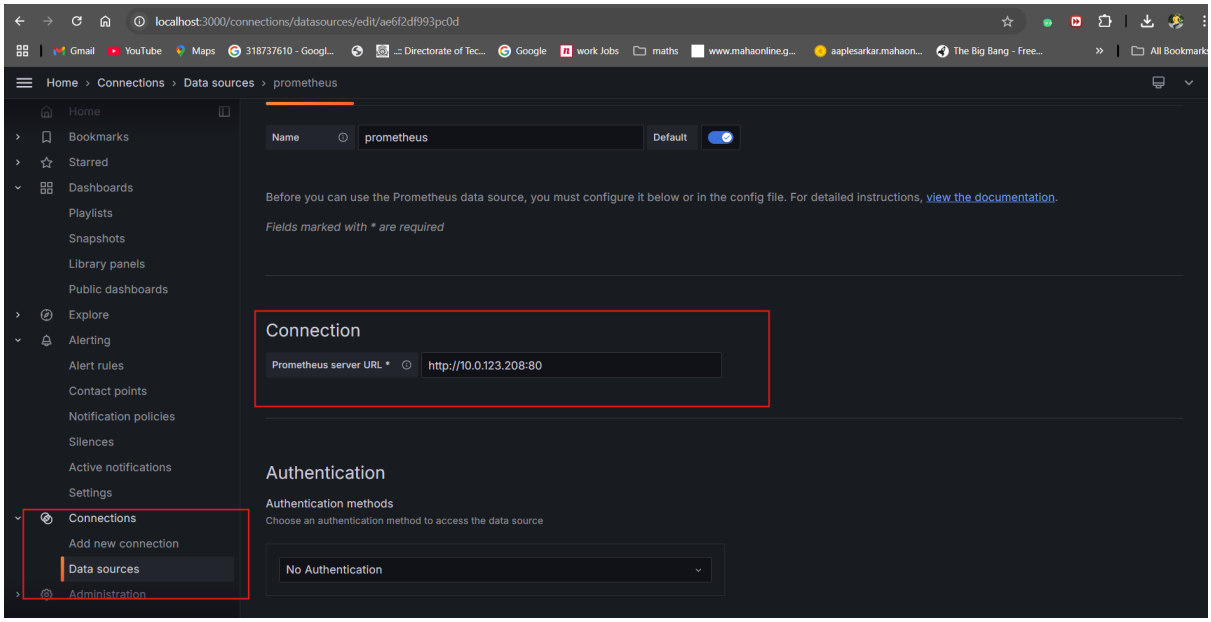

Figure 24: Prometheus Url in Grafana