Assignment 1 Building and Securing a Microservices E-commerce Application

Name : Namash Aggarwal

StudentID : 21023169

Task 1: Set Up Initial Infrastructure

1. Create a Kubernetes Cluster on GKE (or equivalent tool)

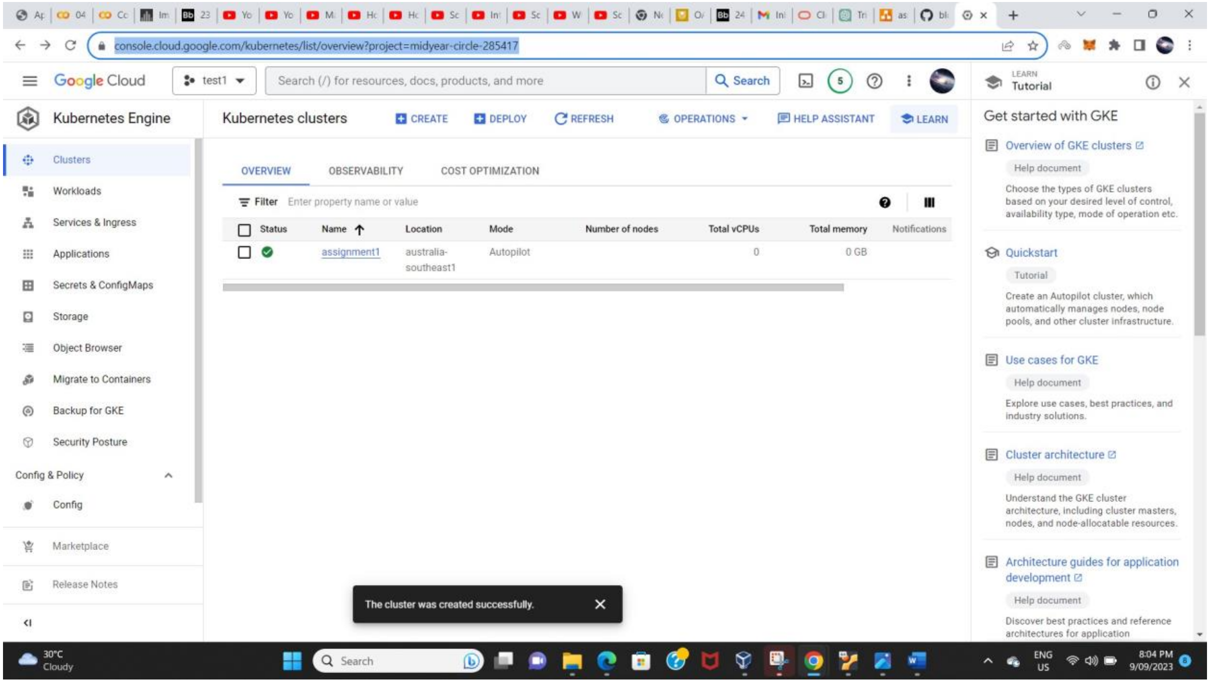
a. Log in to your Google Cloud Console.

b. Navigate to the Kubernetes Engine section and click "Create Cluster."

c. Configure your cluster settings, such as the cluster name, location, and node pool configuration.

d. Choose the desired Kubernetes version.

e. Click "Create" to provision your GKE cluster.



console.cloud.google.com/kubernetes/list/overview?project=midyear-circle-285417

Kubernetes clusters

Status	Name	Location	Mode	Number of nodes	Total vCPUs	Total memory	Notifications
✓	assignment1	australia-southeast1	Autopilot		0	0 GB	

The cluster was created successfully.

Get started with GKE

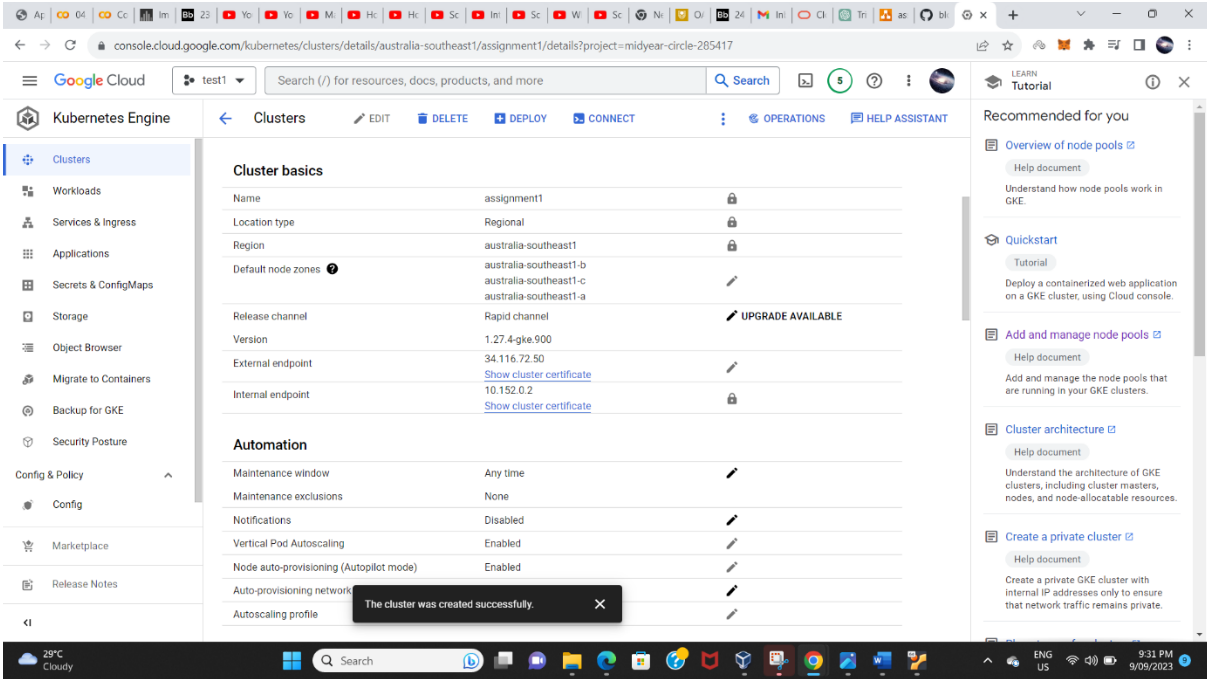
Overview of GKE clusters
Help document
Choose the types of GKE clusters based on your desired level of control, availability type, mode of operation etc.

Quickstart
Tutorial
Create an Autopilot cluster, which automatically manages nodes, node pools, and other cluster infrastructure.

Use cases for GKE
Help document
Explore use cases, best practices, and industry solutions.

Cluster architecture
Help document
Understand the GKE cluster architecture, including cluster masters, nodes, and node-allocatable resources.

Architecture guides for application development
Help document
Discover best practices and reference architectures for application



Cluster basics

Name	assignment1	
Location type	Regional	
Region	australia-southeast1	
Default node zones	australia-southeast1-b australia-southeast1-c australia-southeast1-a	
Release channel	Rapid channel	UPGRADE AVAILABLE
Version	1.27.4-gke.900	
External endpoint	34.116.72.50 Show cluster certificate	
Internal endpoint	10.152.0.2 Show cluster certificate	

Automation

Maintenance window	Any time	
Maintenance exclusions	None	
Notifications	Disabled	
Vertical Pod Autoscaling	Enabled	
Node auto-provisioning (Autopilot mode)	Enabled	
Auto-provisioning network		
Autoscaling profile		

The cluster was created successfully.

Recommended for you

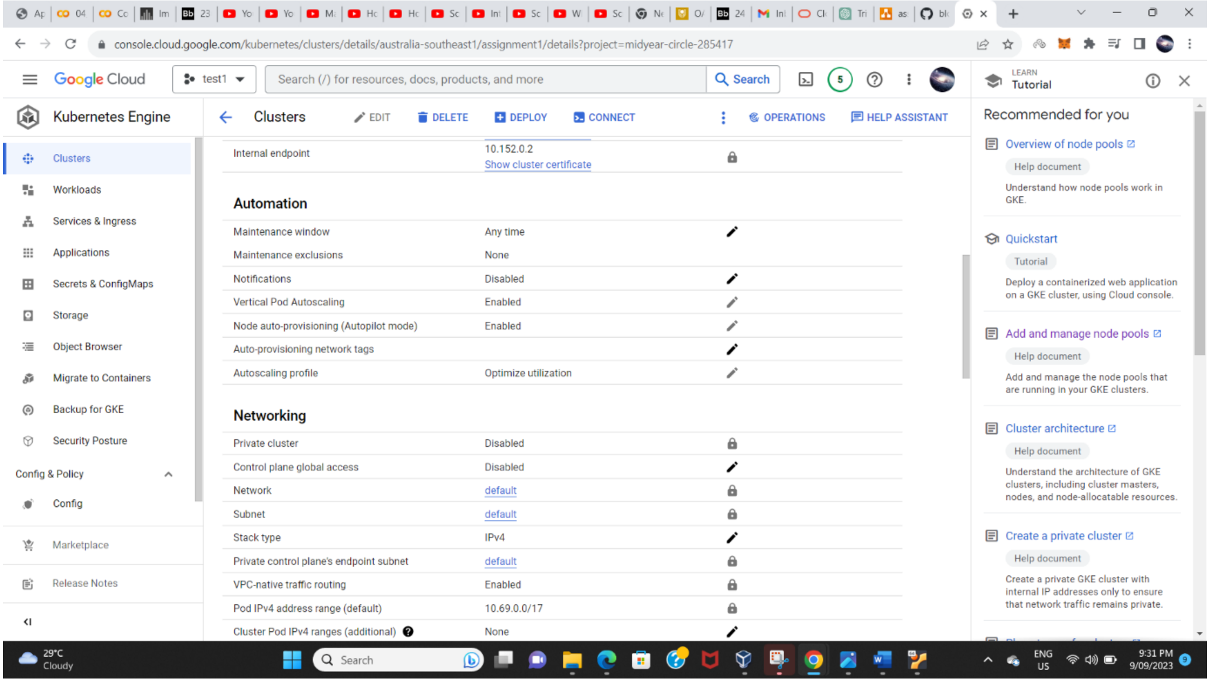
Overview of node pools
Help document
Understand how node pools work in GKE.

Quickstart
Tutorial
Deploy a containerized web application on a GKE cluster, using Cloud console.

Add and manage node pools
Help document
Add and manage the node pools that are running in your GKE clusters.

Cluster architecture
Help document
Understand the architecture of GKE clusters, including cluster masters, nodes, and node-allocatable resources.

Create a private cluster
Help document
Create a private GKE cluster with internal IP addresses only to ensure that network traffic remains private.



Internal endpoint	10.152.0.2 Show cluster certificate	

Automation

Maintenance window	Any time	
Maintenance exclusions	None	
Notifications	Disabled	
Vertical Pod Autoscaling	Enabled	
Node auto-provisioning (Autopilot mode)	Enabled	
Auto-provisioning network tags		
Autoscaling profile	Optimize utilization	

Networking

Private cluster	Disabled	
Control plane global access	Disabled	
Network	default	
Subnet	default	
Stack type	IPv4	
Private control plane's endpoint subnet	default	
VPC-native traffic routing	Enabled	
Pod IPv4 address range (default)	10.69.0.0/17	
Cluster Pod IPv4 ranges (additional)	None	

Recommended for you

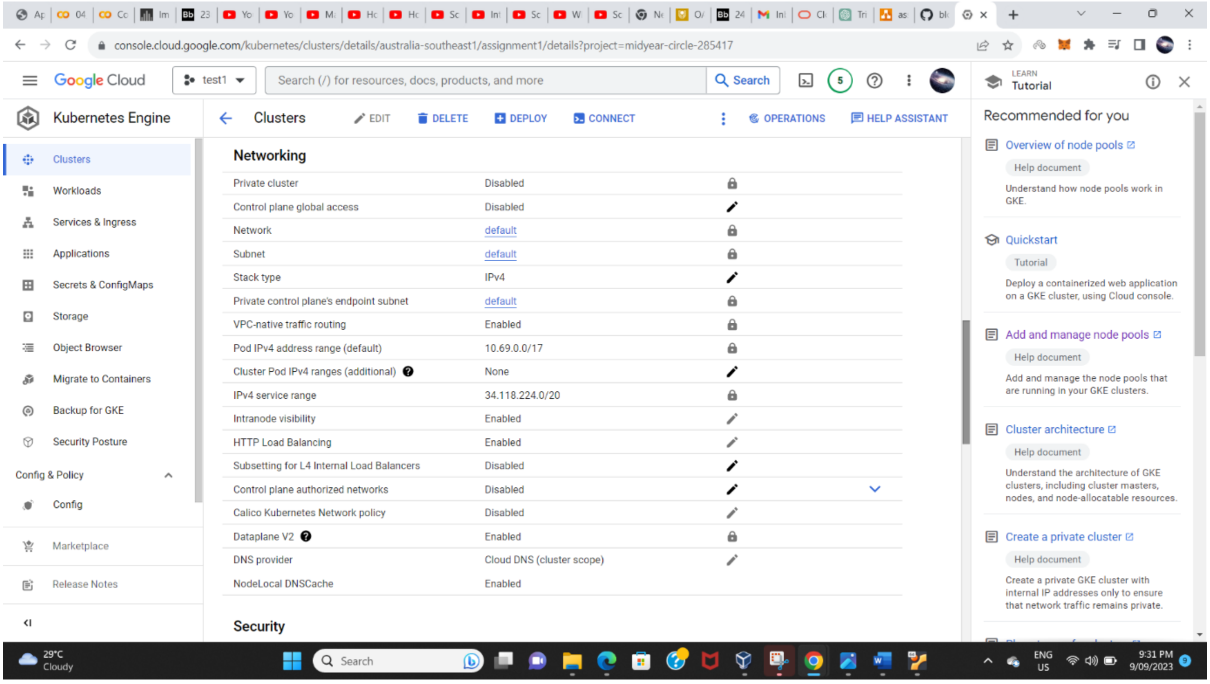
Overview of node pools
Help document
Understand how node pools work in GKE.

Quickstart
Tutorial
Deploy a containerized web application on a GKE cluster, using Cloud console.

Add and manage node pools
Help document
Add and manage the node pools that are running in your GKE clusters.

Cluster architecture
Help document
Understand the architecture of GKE clusters, including cluster masters, nodes, and node-allocatable resources.

Create a private cluster
Help document
Create a private GKE cluster with internal IP addresses only to ensure that network traffic remains private.



console.cloud.google.com/kubernetes/clusters/details/australia-southeast1/assignment1/details?project=midyear-circle-285417

Networking

Private cluster	Disabled	🔒
Control plane global access	Disabled	✎
Network	default	🔒
Subnet	default	🔒
Stack type	IPv4	✎
Private control plane's endpoint subnet	default	🔒
VPC-native traffic routing	Enabled	🔒
Pod IPv4 address range (default)	10.69.0.0/17	🔒
Cluster Pod IPv4 ranges (additional) ?	None	✎
IPv4 service range	34.118.224.0/20	🔒
Intranode visibility	Enabled	✎
HTTP Load Balancing	Enabled	✎
Subsetting for L4 Internal Load Balancers	Disabled	✎
Control plane authorized networks	Disabled	✎
Calico Kubernetes Network policy	Disabled	✎
Dataplane V2 ?	Enabled	🔒
DNS provider	Cloud DNS (cluster scope)	✎
NodeLocal DNSCache	Enabled	

Security

Recommended for you

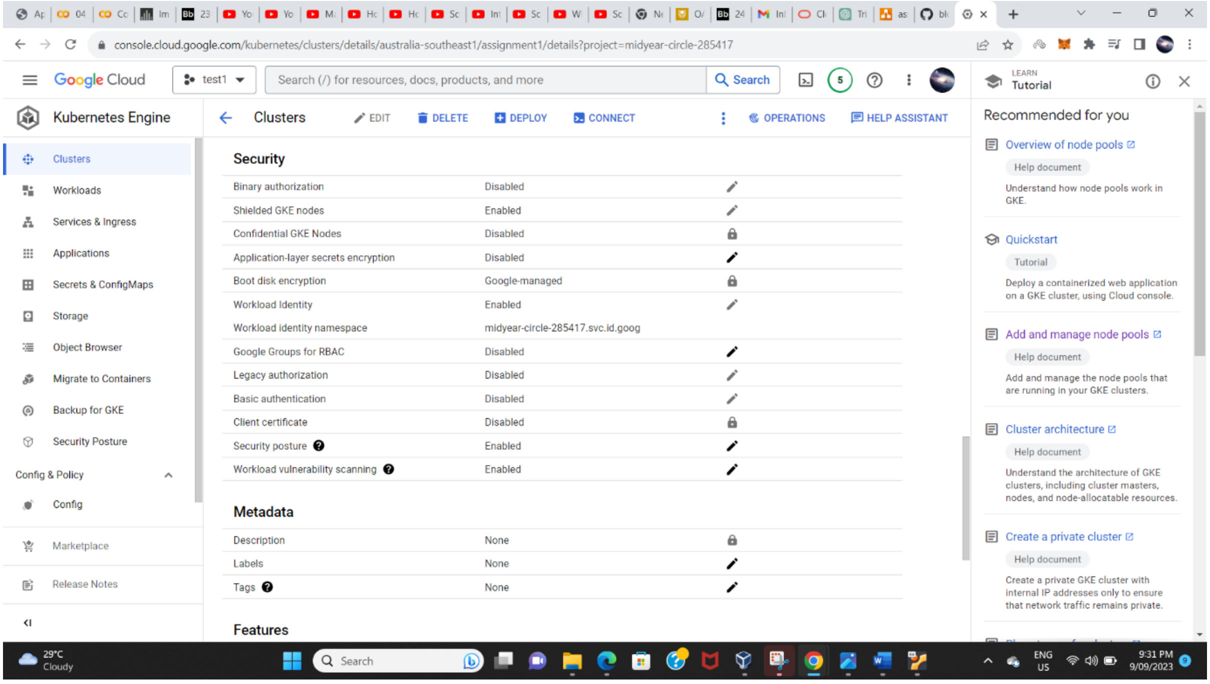
Overview of node pools
Help document
Understand how node pools work in GKE.

Quickstart
Tutorial
Deploy a containerized web application on a GKE cluster, using Cloud console.

Add and manage node pools
Help document
Add and manage the node pools that are running in your GKE clusters.

Cluster architecture
Help document
Understand the architecture of GKE clusters, including cluster masters, nodes, and node-allocatable resources.

Create a private cluster
Help document
Create a private GKE cluster with internal IP addresses only to ensure that network traffic remains private.



console.cloud.google.com/kubernetes/clusters/details/australia-southeast1/assignment1/details?project=midyear-circle-285417

Security

Binary authorization	Disabled	✎
Shielded GKE nodes	Enabled	✎
Confidential GKE Nodes	Disabled	🔒
Application-layer secrets encryption	Disabled	✎
Boot disk encryption	Google-managed	🔒
Workload Identity	Enabled	✎
Workload identity namespace	midyear-circle-285417.svc.id.goog	
Google Groups for RBAC	Disabled	✎
Legacy authorization	Disabled	✎
Basic authentication	Disabled	✎
Client certificate	Disabled	🔒
Security posture ?	Enabled	✎
Workload vulnerability scanning ?	Enabled	✎

Metadata

Description	None	🔒
Labels	None	✎
Tags ?	None	✎

Features

Recommended for you

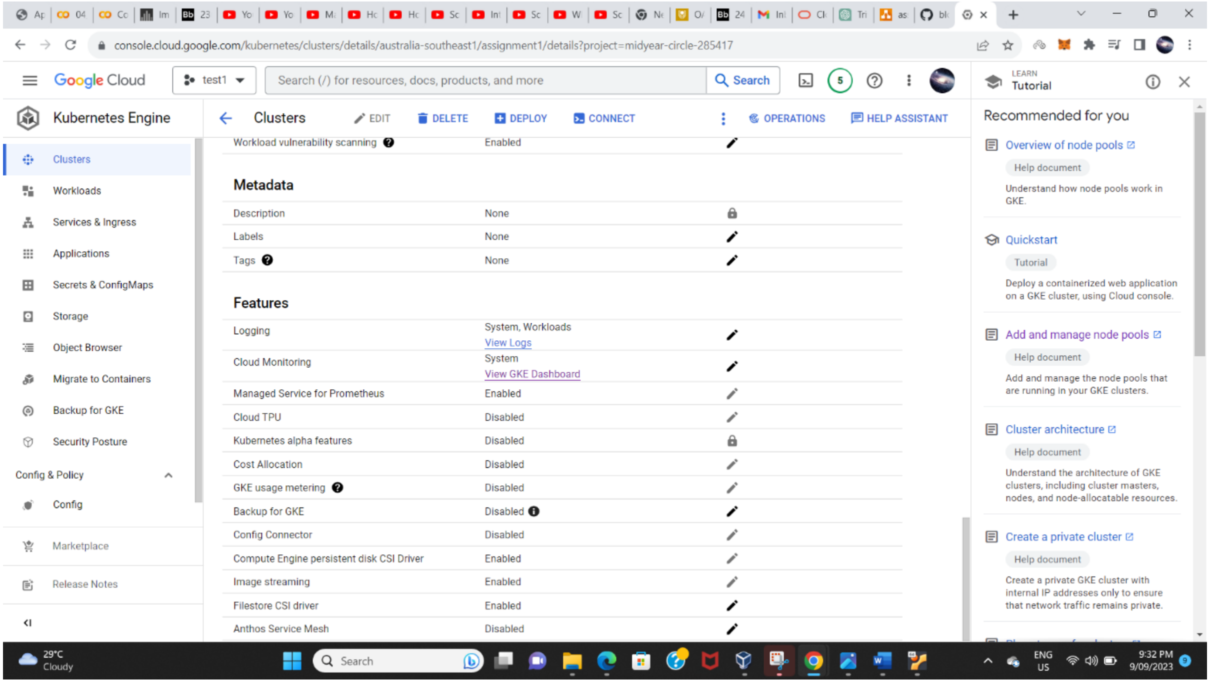
Overview of node pools
Help document
Understand how node pools work in GKE.

Quickstart
Tutorial
Deploy a containerized web application on a GKE cluster, using Cloud console.

Add and manage node pools
Help document
Add and manage the node pools that are running in your GKE clusters.

Cluster architecture
Help document
Understand the architecture of GKE clusters, including cluster masters, nodes, and node-allocatable resources.

Create a private cluster
Help document
Create a private GKE cluster with internal IP addresses only to ensure that network traffic remains private.



console.cloud.google.com/kubernetes/clusters/details/australia-southeast1/assignment1/details?project=midyear-circle-285417
Google Cloud
test1
Search (/) for resources, docs, products, and more
Search
Kubernetes Engine
Clusters
Workloads
Services & Ingress
Applications
Secrets & ConfigMaps
Storage
Object Browser
Migrate to Containers
Backup for GKE
Security Posture
Config & Policy
Config
Marketplace
Release Notes
Clusters
EDIT
DELETE
DEPLOY
CONNECT
OPERATIONS
HELP ASSISTANT
LEARN
Tutorial
Recommended for you
Overview of node pools
Help document
Understand how node pools work in GKE.
Quickstart
Tutorial
Deploy a containerized web application on a GKE cluster, using Cloud console.
Add and manage node pools
Help document
Add and manage the node pools that are running in your GKE clusters.
Cluster architecture
Help document
Understand the architecture of GKE clusters, including cluster masters, nodes, and node-allocatable resources.
Create a private cluster
Help document
Create a private GKE cluster with internal IP addresses only to ensure that network traffic remains private.
29°C
Cloudy
ENG
US
9:32 PM
9/09/2023

Workload vulnerability scanning	Enabled

Metadata

Description	None
Labels	None
Tags	None

Features

Logging	System, Workloads View Logs
Cloud Monitoring	System View GKE Dashboard
Managed Service for Prometheus	Enabled
Cloud TPU	Disabled
Kubernetes alpha features	Disabled
Cost Allocation	Disabled
GKE usage metering	Disabled
Backup for GKE	Disabled
Config Connector	Disabled
Compute Engine persistent disk CSI Driver	Enabled
Image streaming	Enabled
Filestore CSI driver	Enabled
Anthos Service Mesh	Disabled

2. Install and configure kubectl to manage your Kubernetes cluster.

a. After creating the GKE cluster, you will need to configure your local environment to use kubectl to interact with this cluster.

b. In the Google Cloud Console, navigate to the "Kubernetes Engine" > "Clusters" section and click the "Connect" button next to your cluster.

c. Follow the instructions to authenticate kubectl with your GKE cluster.

-- curl https://sdk.cloud.google.com | bash

-- exec -l $SHELL

-- gcloud init

For Kubectl

- gcloud components install kubectl

Authenticate Kubectl

- gcloud container clusters get-credentials assignment1 --australia-southeast1 --project 21023169

Check Configurations

- kubectl config current-context

Check connections

- kubectl get nodes

3. Set up a private GitHub repository to store your project files.

a. Log in to your GitHub account (or create one if you don't have it).

b. Click on the '+' icon at the top right corner of the GitHub dashboard and select "New repository."
c. Choose a meaningful repository name for your ISEC6000 Secure DevOps project.

d. Select "Public" for the repository visibility.

e. You need to add a description and choose whether to initialize the repository with a README. (Bonus marks if you have a proper README file.

f. Click "Create repository."

g. Push the initial setup code to the repository.

https://github.com/blousy/-isec6000-assignment1--task1.git

Task 2: Microservices Architecture and Deployment

1. Begin by acquainting yourself with the core projects and their purposes:

a. Saleor API: Explore the functionalities at https://github.com/saleor/saleor .

b. Saleor storefront: Understand the frontend mechanics at https://github.com/saleor/react-storefront .

c. Saleor dashboard: Dive into the dashboard intricacies at https://github.com/saleor/saleor-dashboard .

d. Saleor platform: Access the repository at https://github.com/saleor/saleorplatform , which contains essential Docker Compose elements for configuring, building, and executing Saleor components. Note that this repository references the three aforementioned repos using Git submodules.

2. Create a personal account on Github.com and proceed to fork the Saleor platform repository.

3. Follow the step-by-step guidelines outlined in the Saleor platform repository to effectively run a Saleor stack enriched with sample data.

4. Tailor the Compose file to ensure optimal functionality:

a. Configure the React Storefront to operate on port 3009.

b. Assign port 9003 for the Saleor Dashboard. c. Initiate the stack and verify the successful launch of all services: o Saleor React Storefront: Accessible at http://:3009. o Saleor Dashboard: Reachable via http://:9003.

5. Commit your modifications and push them to the forked repository, appending the tag isec6000-assignment1.

https://github.com/blousy/-isec6000-assignment1--task2.git

Task 3

Implementing Security Measures

1. **Container Security: a. Ensure secure configuration of containers**.

Ensuring the secure configuration of Docker containers on an Ubuntu system is an essential step to protect the applications and data. Here are some recommended best practices:

1**. Keep Everything Updated**

- Regularly update your Ubuntu system, Docker engine, and the containers themselves to get the latest security patches.

```
Sudo apt update && sudo apt upgrade
```

2**. Use Verified Images**

Only use Docker images from trusted sources like Docker Hub's official repositories or verified publishers.

3. **Least Privilege Principle**

Run containers with the least privileges possible (avoid running containers as root). You can specify a user while starting the container with the `-u` or `--user` option.

```
Sudo docker run --user nobody nginx
```

4**. Immutability and Read-Only**

Where possible, set containers to be read-only by adding `--read-only` flag when you run them. This prevents any changes to the file system during runtime.

```
Sudo docker run --read-only nginx
```

5. **Disable Inter-Container Communication**

- Isolate containers using Docker’s networking capabilities (`--icc=false` flag).

```
  Sudo docker run --icc=false nginx
```

6. **Use Specific Host Interfaces**

- Don't expose your container to every network interface. Specify the IP address that the container will bind to.

```
   docker run -p 127.0.0.1:$HOST_PORT:$CONTAINER_PORT nginx
```

7. **Logging and Monitoring**

- Implement robust logging mechanisms to monitor container activity. Tools like ELK stack, Grafana, or Prometheus can be useful.

8. **Limit Resources**

- Use `--cpus` and `--memory` flags to limit container resources, preventing DoS attacks.

```
   Sudo docker run --cpus=".5" --memory="256m" nginx
```

9. **Scan for Vulnerabilities**

- Tools like Trivy, Clair, or Anchore can scan Docker images for known vulnerabilities.

```
  Sudo trivy image nginx:latest
```

10. **Secure Docker Daemon**

- Ensure the Docker daemon is securely configured. You can use options in `/etc/docker/daemon.json` to disable insecure features.

11. **Use Docker Compose for Config**

- If possible, use Docker Compose to manage configurations in a `docker-compose.yml` file. This will make it easier to manage secure configurations across multiple containers.

12**. Filesystem and Volumes**

- Be cautious when mounting host directories as Docker volumes. Ensure that they are properly secured with the right permissions and access controls.

By implementing these security best practices, you'll be taking significant steps to ensure that your Docker containers are as secure as possible on your Ubuntu system.

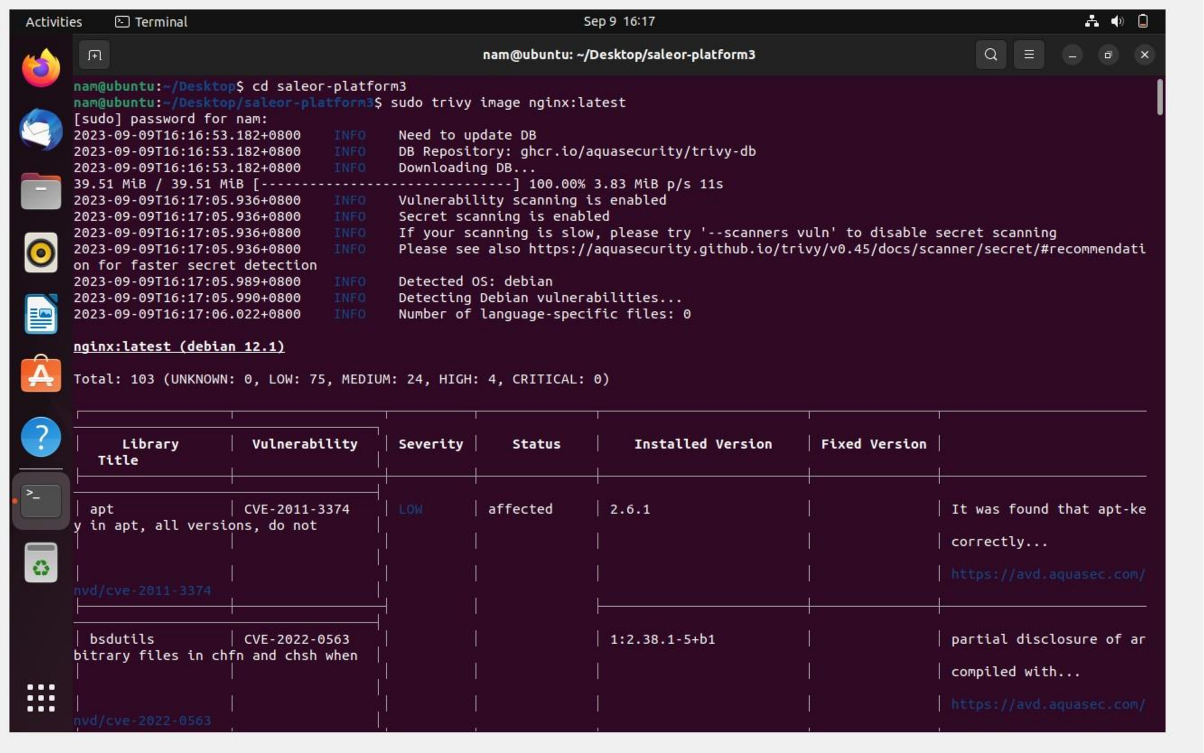
b. Implement container image vulnerability scanning using tools like Trivy.

I installed Trivy in my Ubuntu system using the commands:

```
sudo apt-get install wget apt-transport-https gnupg lsb-release

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 -a /etc/apt/sources.list.d/trivy.list

sudo apt-get update

sudo apt-get install trivy
```

Secondly, I scanned the image for any vulnerabilities ;

```
Sudo trivy image nginx:latest
```



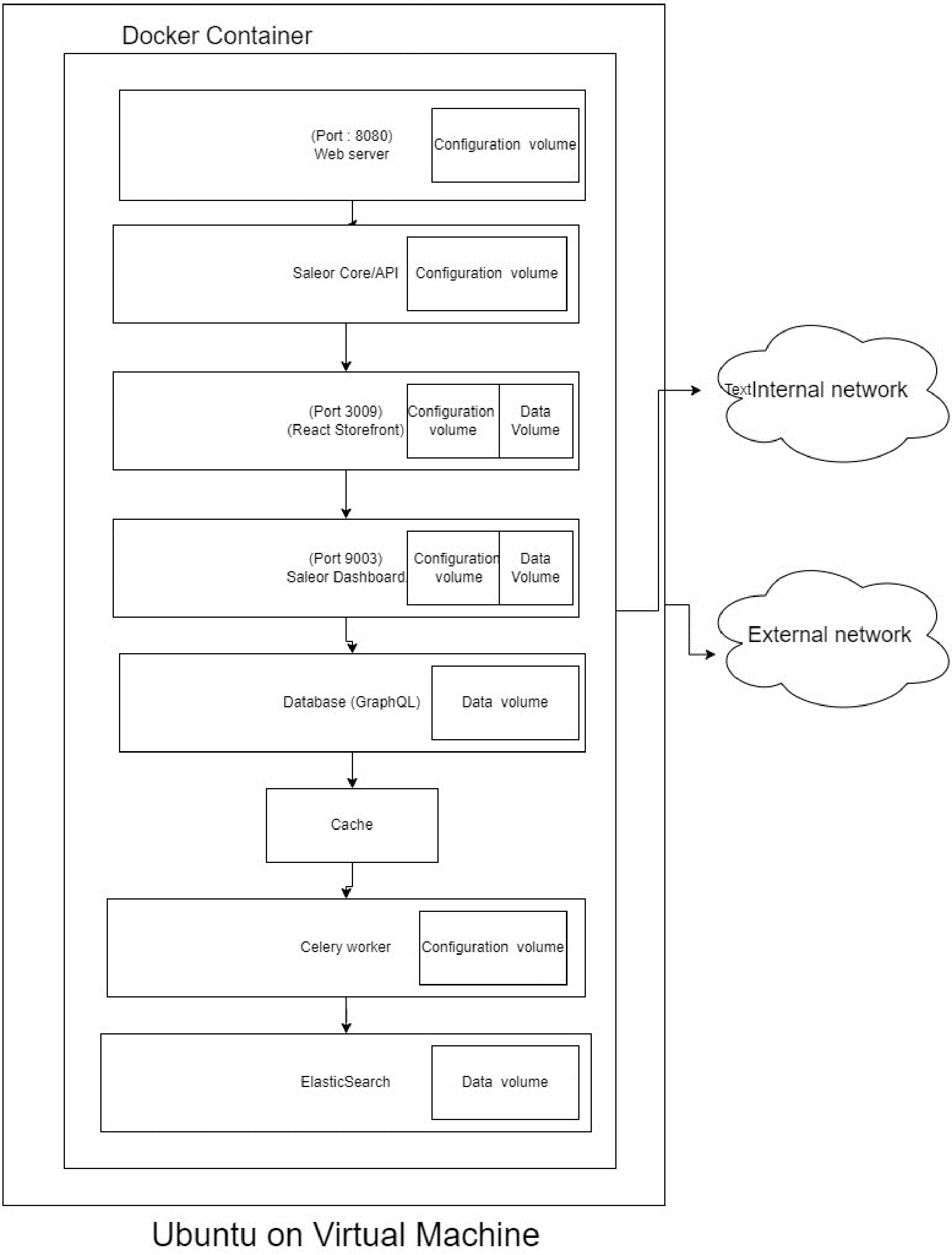
```
nam@ubuntu:~/Desktop$ cd saleor-platform3
nam@ubuntu:~/Desktop/saleor-platform3$ sudo trivy image nginx:latest
[sudo] password for nam:
2023-09-09T16:16:53.182+0800    INFO    Need to update DB
2023-09-09T16:16:53.182+0800    INFO    DB Repository: ghcr.io/aquasecurity/trivy-db
2023-09-09T16:16:53.182+0800    INFO    Downloading DB...
39.51 MiB / 39.51 MiB [------------------------------] 100.00% 3.83 MiB p/s 11s
2023-09-09T16:17:05.936+0800    INFO    Vulnerability scanning is enabled
2023-09-09T16:17:05.936+0800    INFO    Secret scanning is enabled
2023-09-09T16:17:05.936+0800    INFO    If your scanning is slow, please try '--scanners vuln' to disable secret scanning
2023-09-09T16:17:05.936+0800    INFO    Please see also https://aquasecurity.github.io/trivy/v0.45/docs/scanner/secret/#recommendati
on for faster secret detection
2023-09-09T16:17:05.989+0800    INFO    Detected OS: debian
2023-09-09T16:17:05.990+0800    INFO    Detecting Debian vulnerabilities...
2023-09-09T16:17:06.022+0800    INFO    Number of language-specific files: 0

nginx:latest (debian 12.1)

Total: 103 (UNKNOWN: 0, LOW: 75, MEDIUM: 24, HIGH: 4, CRITICAL: 0)

┌──────────────────┬──────────────────┬──────────┬──────────────┬───────────────────────┬──────────────┬──────────────────────────
│     Library      │  Vulnerability   │ Severity │    Status    │   Installed Version   │ Fixed Version│
   Title
├──────────────────┼──────────────────┼──────────┼──────────────┼───────────────────────┼──────────────┼──────────────────────────
│ apt              │ CVE-2011-3374    │ LOW      │ affected     │ 2.6.1                 │              │ It was found that apt-ke
y in apt, all versions, do not
│                  │                  │          │              │                       │              │ correctly...
│                  │                  │          │              │                       │              │ https://avd.aquasec.com/
nvd/cve-2011-3374
├──────────────────┼──────────────────┼──────────┤              ├───────────────────────┼──────────────┼──────────────────────────
│ bsdutils         │ CVE-2022-0563    │          │              │ 1:2.38.1-5+b1         │              │ partial disclosure of ar
bitrary files in chfn and chsh when
│                  │                  │          │              │                       │              │ compiled with...
│                  │                  │          │              │                       │              │ https://avd.aquasec.com/
nvd/cve-2022-0563
```

Task 4

Architecture Visualization [20]

1. Present an architecture diagram that encapsulates the essential functionalities of each Saleor service, accompanied by their interactions within the stack. You have the creative freedom to choose diagram style and tools that best convey your design, with the understanding that clarity and conciseness are pivotal. The architecture diagram should encompass the following aspects: • Illustrate the volumes and networks utilized by different Saleor services. • Depict how the different Saleor services communicate with each other. • Highlight the exposed ports associated with each container.

Services


Docker Container
(Port : 8080)
Web server
Configuration volume
Saleor Core/API
Configuration volume
(Port 3009)
(React Storefront)
Configuration volume
Data Volume
(Port 9003)
Saleor Dashboard
Configuration volume
Data Volume
Database (GraphQL)
Data volume
Cache
Celery worker
Configuration volume
ElasticSearch
Data volume
Ubuntu on Virtual Machine
TextInternal network
External network