List of Figures

Figure 1: Setting up cluster name, location and node pool 4
Figure 2: Cluster created 4
Figure 3: Authenticated kubectl with cluster 4
Figure 4: Initial Git Repository 5
Figure 5: Cloning the git repo and using docker compose 6
Figure 6: Using docker compose up to run the application 6
Figure 7: Changing port address from 3000 to 3009 and pushed to github repository 7
Figure 8: React storefront having error by default 7
Figure 9: Port address of dashboard changed from 9000 to 9003 and pushed to github repository 8
Figure 10: Saleor Dashboard page 8
Figure 11: Dashboard page after logging in 9
Figure 12: Tag for Task 1 9
Figure 13: Tag for Task 2 10
Figure 14: Image Scan for Saleor Dashboard 11
Figure 15:Image Scan for React Storefront 12
Figure 16: Vulnerability in react storefront 12
Figure 17: Vulnerability in react storefront 2 13
Figure 18: Architecture Diagram for Saleor service 13

Table of Contents

1. Introduction *3*

2. Task 1: Set Up Initial Infrastructure *3*

3. Task 2: Microservices Architecture and Deployment *5*

4. Task 3: Implementing Security Measures *10*

4.1. Ensuring Secure Configuration of containers **10**

4.2. Implementation of container image vulnerability **11**

5. Task 4: Architecture Visualization *13*

6. Conclusion *14*

Reference *15*

1. Introduction

Being a part of cutting-edge technology team who is responsible for designing, deploying, and securing a modern e-commerce application and application is designed and deployed that will revolutionize online shopping by incorporating the latest advancements in not only technology but in security and DevOps practices as well.

In this project Saleor application suite is skillfully constructed, configured, and deployed on a linux server. Saleor is an open-source ecommerce platform that uses a Python API backend, React Javascript in the frontend and PostgresSQL database in the backend. The creation of cluster in the google and its deployment is done in the first phase and in the second phase the microservices architecture deployment is done. The tasks done are sub divided in the two phases Task 1 and Task 2 respectively.

2. Task 1: Set Up Initial Infrastructure

Git Link: https://github.com/diwashc/ISEC6000-Secure-DevOps

In this phase a Kubernetes cluster is created on GKE. For this at first logging in google cloud console is required. Then navigating inside Kubernetes Engine section using the GUI after clicking “Create Cluster” a new cluster is created as shown in figure 1. Then as shown in figure 2 we can see the cluster name, location, number of nodes, total CPU and total memory of the cluster that is created.

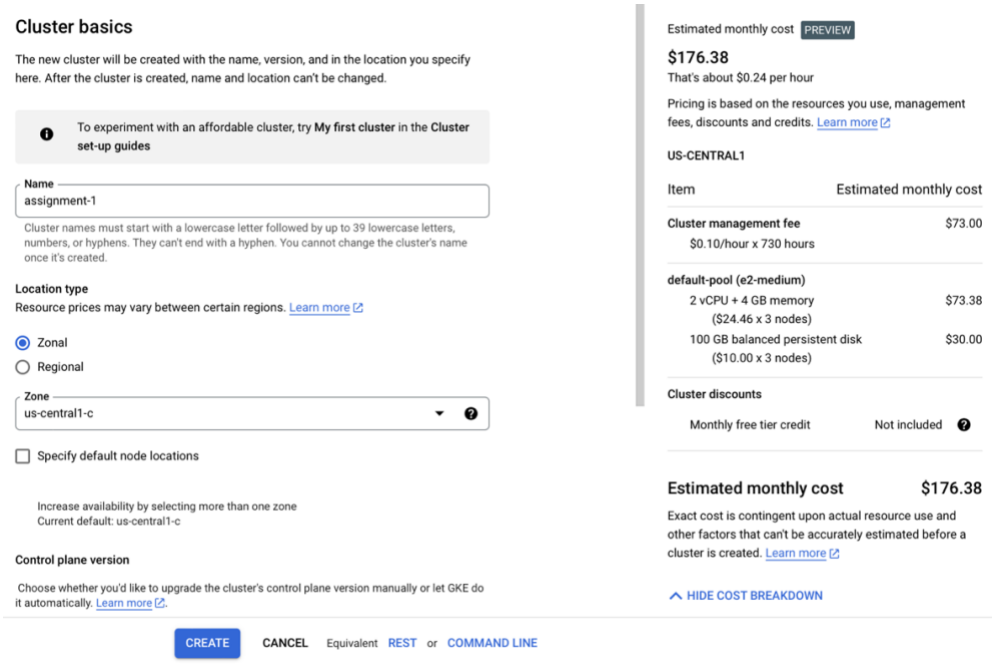

Figure 1: Setting up cluster name, location and node pool

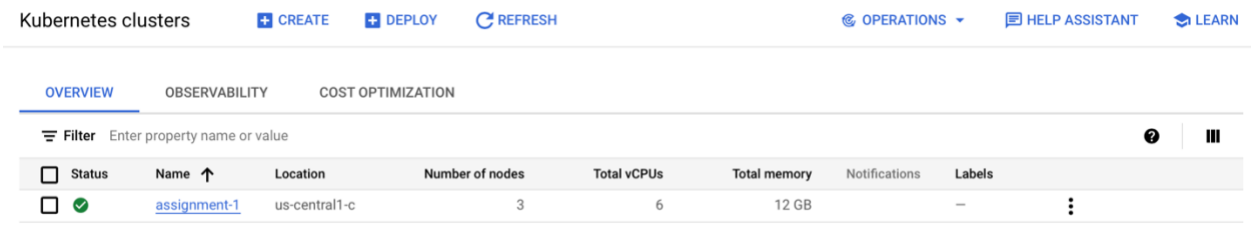

Figure 2: Cluster created

To configure the local environment to use kubectl to interact with the cluster we use the command "gcloud container clusters get-credentials CLUSTERNAME –zone us-central1-c –project PROJECT NAME". As shown in figure 3 kubeconfig entry was generated for assignment-1.

```
diwashc@cloudshell:~ (isec6000-397804)$ gcloud container clusters get-credentials assignment-1 --zone us-central1-c --project isec6000-397804
Fetching cluster endpoint and auth data.
kubeconfig entry generated for assignment-1.
diwashc@cloudshell:~ (isec6000-397804)$
```

Figure 3: Authenticated kubectl with cluster

Version control is used to track the files and helps to go back in previous working version if ever needed in the future. (Betterexplained, 2023) ISEC6000-Secure-Devops repository is created to keep track of the tasks done in the project. The readme.md file in the repository consists of the steps and requirements required for the project.

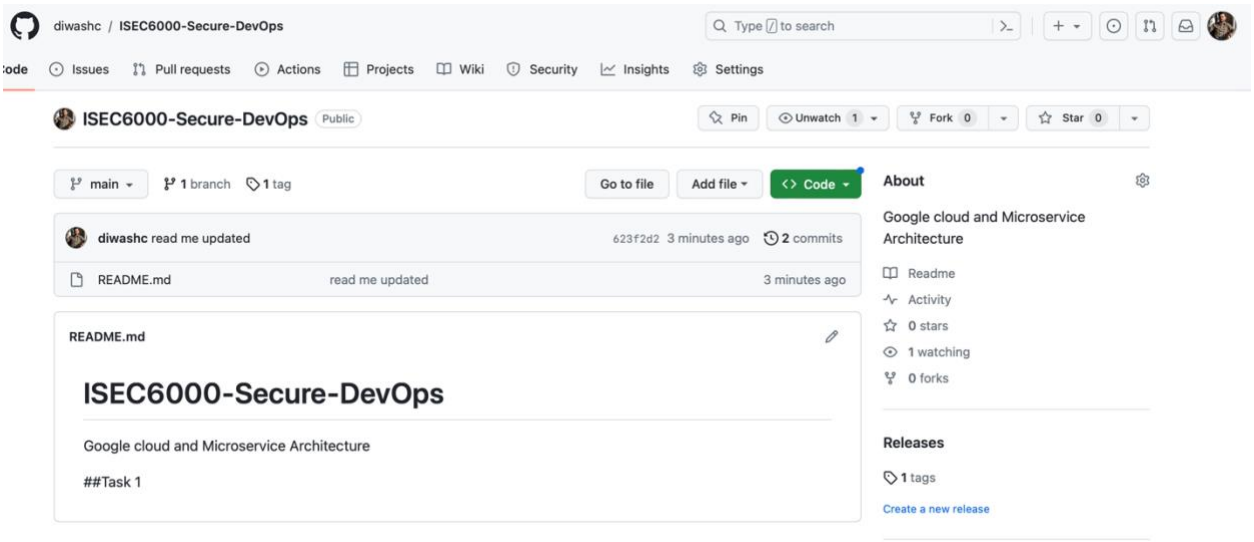

Figure 4: Initial Git Repository

3. Task 2: Microservices Architecture and Deployment

Git Link Saleor Platform: https://github.com/diwashc/saleor-platform

Git Link React StoreFront: https://github.com/diwashc/react-storefront

For task 2 Saleor and version control is used. Saleor is a GraphQL API that provides the backend for the frontend applications. (Saleor, 2023) The repositories for Saleor application to be used is publicly available in github and initially to use it Saleor platform is forked into our github account so that further changes can be maintained using the version control. React-storefront is also forked to change the port address for saleor react storefront.

```
diwashchand@Diwashs Assignment % git clone git@github.com:diwashc/saleor-platform.git
Cloning into 'saleor-platform'...
remote: Enumerating objects: 501, done.
remote: Counting objects: 100% (181/181), done.
remote: Compressing objects: 100% (114/114), done.
remote: Total 501 (delta 100), reused 134 (delta 61), pack-reused 320
Receiving objects: 100% (501/501), 142.15 KiB | 353.00 KiB/s, done.
Resolving deltas: 100% (260/260), done.
diwashchand@Diwashs Assignment % ls
ISEC6000-Secure-DevOps          SecureDevOps-Assignment1.pdf
P1_21255001.docx                saleor-platform
README.md
diwashchand@Diwashs Assignment % cd saleor-platform
diwashchand@Diwashs saleor-platform % docker compose build
[+] Building 0.0s (0/0)                                  docker:desktop-linux
diwashchand@Diwashs saleor-platform % docker compose run --rm api python3 manage.py migrate
[+] Creating 5/2
 ✔ Network saleor-platform_default              Created                    0.0s
 ✔ Network saleor-platform_saleor-backend-tier  Created                    0.0s
 ✔ Container saleor-platform-redis-1            Created                    0.0s
 ✔ Container saleor-platform-jaeger-1           Created                    0.0s
 ✔ Container saleor-platform-db-1               Created                    0.0s
[+] Running 3/3
 ✔ Container saleor-platform-jaeger-1  Started                             0.4s
 ✔ Container saleor-platform-db-1      Started                             0.3s
 ✔ Container saleor-platform-redis-1   Started                             0.3s
2023-09-07 04:06:36,338 WARNING saleor.core.jwt_manager RSA_PRIVATE_KEY is missing. Using temporary key for local development with DEBUG
Operations to perform:
  Apply all migrations: account, app, attribute, auth, channel, checkout, contenttypes, core, csv, discount, django_celery_beat, django_
 menu, order, page, payment, permission, plugins, product, schedulers, shipping, site, sites, tax, thumbnail, warehouse, webhook
Running migrations:
  No migrations to apply.
diwashchand@Diwashs saleor-platform % docker compose run --rm api python3 manage.py populatedb --createsuperuser
[+] Creating 3/0
 ✔ Container saleor-platform-redis-1   Running                             0.0s
 ✔ Container saleor-platform-jaeger-1  Running                             0.0s
```

Figure 5: Cloning the git repo and using docker compose

```
diwashchand@Diwashs saleor-platform % docker compose up
[+] Running 7/7
 ✔ Container saleor-platform-dashboard-1  Created                          0.1s
 ✔ Container saleor-platform-mailpit-1    Created                          0.0s
 ✔ Container saleor-platform-redis-1      Running                          0.0s
 ✔ Container saleor-platform-jaeger-1     Running                          0.0s
 ✔ Container saleor-platform-db-1         Running                          0.0s
 ✔ Container saleor-platform-api-1        Created                          0.0s
 ✔ Container saleor-platform-worker-1     Created                          0.0s
```

Figure 6: Using docker compose up to run the application

As shown in figure 5 “docker compose build” is used to build the application after cloning the saleor platform repository. “docker compose run --rm api python3 manage.py migrate” is used to apply Django migrations. “docker compose run --rm api python3 manage.py populatedb --createsuperuser” is used to populate the database with example data and create the admin user. The dashboard can be accessed by using admin@example.com as username and admin as password. Finally “docker compose up” runs the application.

Now to configure the react storefront to operate on port 3009 port address was changed in docker-compose.yml and pushed to the repository.

Commits on Sep 7, 2023

changed port of react storefront

diwashc committed 2 days ago

Showing **1 changed file** with **2 additions** and **2 deletions**.

4 docker-compose.yml

```
@@ -21,10 +21,10 @@ services:
21 21        dockerfile: Dockerfile.base
22 22        args:
23 23          - SALEOR_API_URL=http://localhost:8000/graphql/ # handled by nginx
24    -        - STOREFRONT_URL=http://localhost:3000
   24 +        - STOREFRONT_URL=http://localhost:3009
25 25          - CHECKOUT_APP_URL=http://localhost:3001
26 26          - CHECKOUT_STOREFRONT_URL=http://localhost:3001/checkout-spa/
27 27          - CLOUD_DEPLOYMENT_URL=
28 28      ports:
29    -      - 3000:3000
   29 +      - 3009:3000
30 30      command: sh -c '(nginx &) && pnpm turbo run start --filter=storefront...'
```

Figure 7: Changing port address from 3000 to 3009 and pushed to github repository

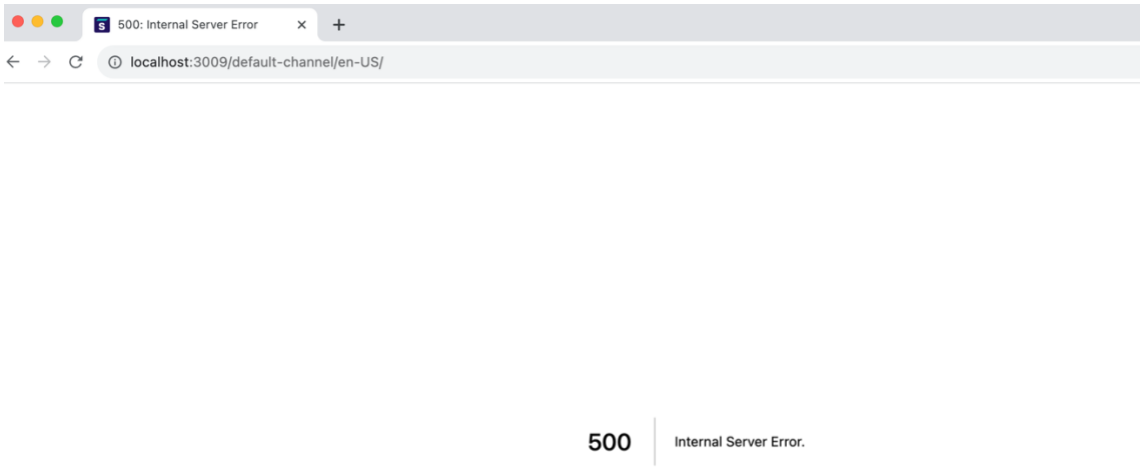

Figure 8: React storefront having error by default

By default, the port address of 3000 i.e storefront_url had 500 error so there was the same error even when changing the port address.

Now, for the dashboard port number 9000 was changed to 9009.

```
   7     7          - 8000:8000
   8     8        restart: unless-stopped
         @@ -22,13 +22,13 @@ services:
  22    22          - backend.env
  23    23        environment:
  24    24          - JAEGER_AGENT_HOST=jaeger
  25        -       - DASHBOARD_URL=http://localhost:9000/
        25  +       - DASHBOARD_URL=http://localhost:9003/
  26    26          - ALLOWED_HOSTS=localhost,api
  27    27
  28    28      dashboard:
  29        -     image: ghcr.io/saleor/saleor-dashboard:3.15.2
        29  +     image: ghcr.io/saleor/saleor-dashboard:3.14.4
  30    30        ports:
  31        -       - 9000:80
        31  +       - 9003:80
  32    32        restart: unless-stopped
  33    33
```

Figure 9: Port address of dashboard changed from 9000 to 9003 and pushed to github repository

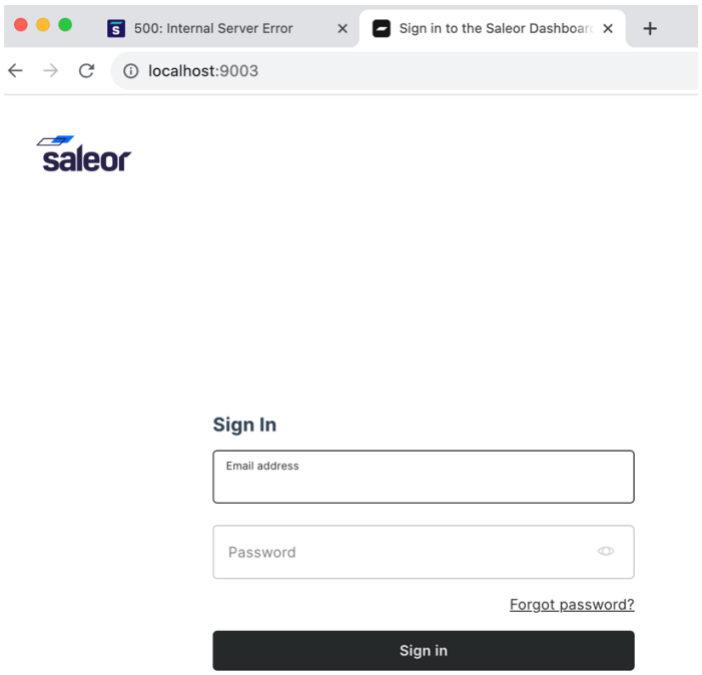

Figure 10: Saleor Dashboard page

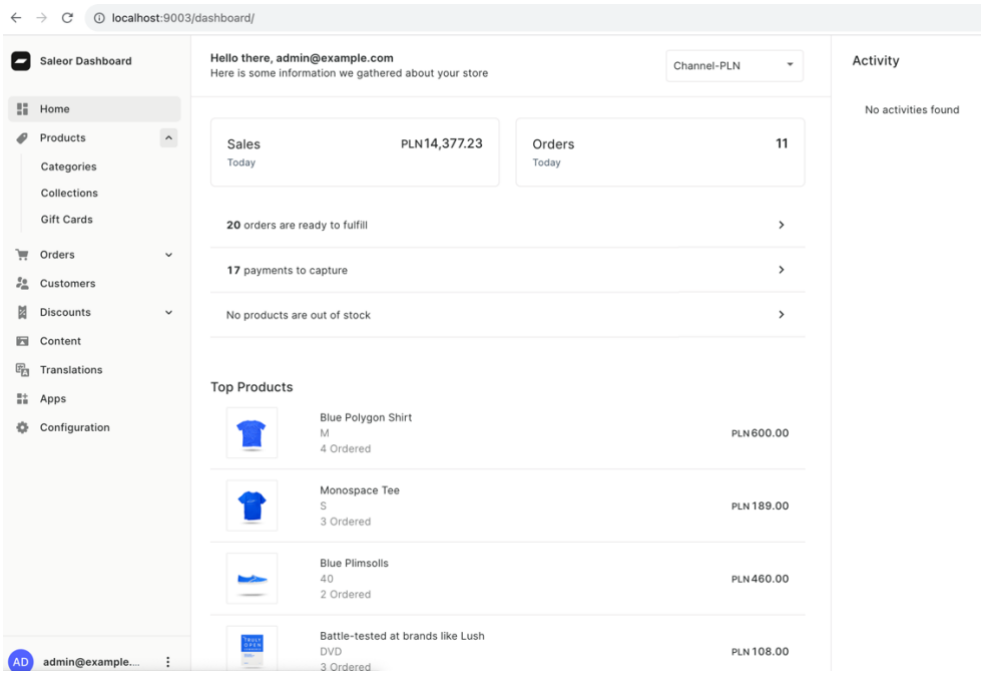

Figure 11: Dashboard page after logging in

Using admin@example.com as username and admin as password the dashboard page as shown in figure 11 is displayed.

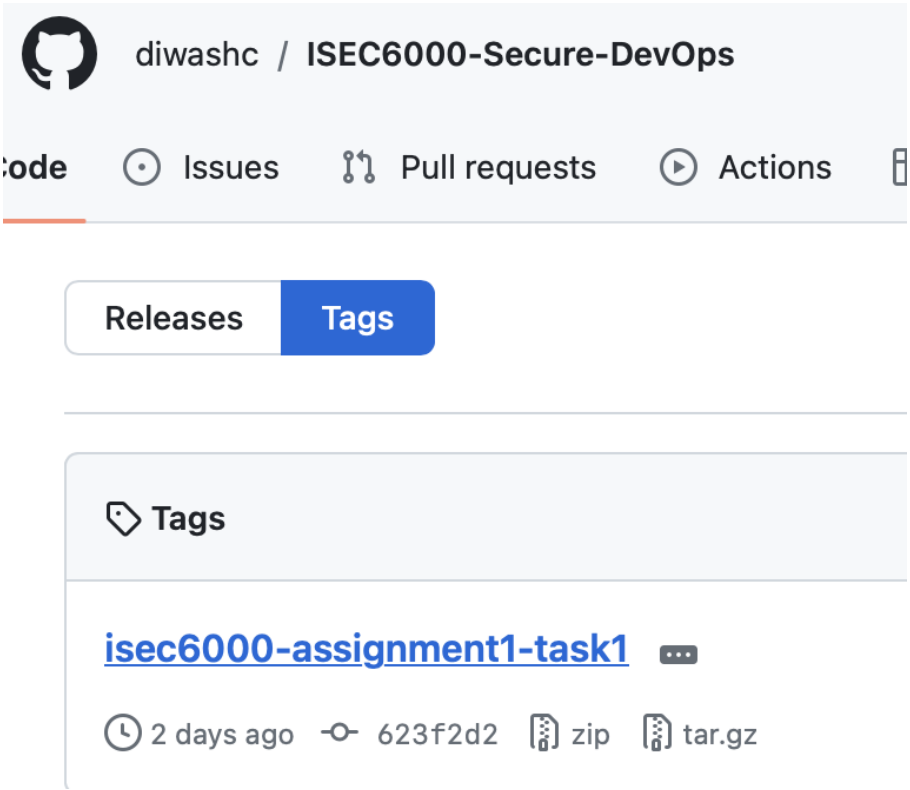

Figure 12: Tag for Task 1

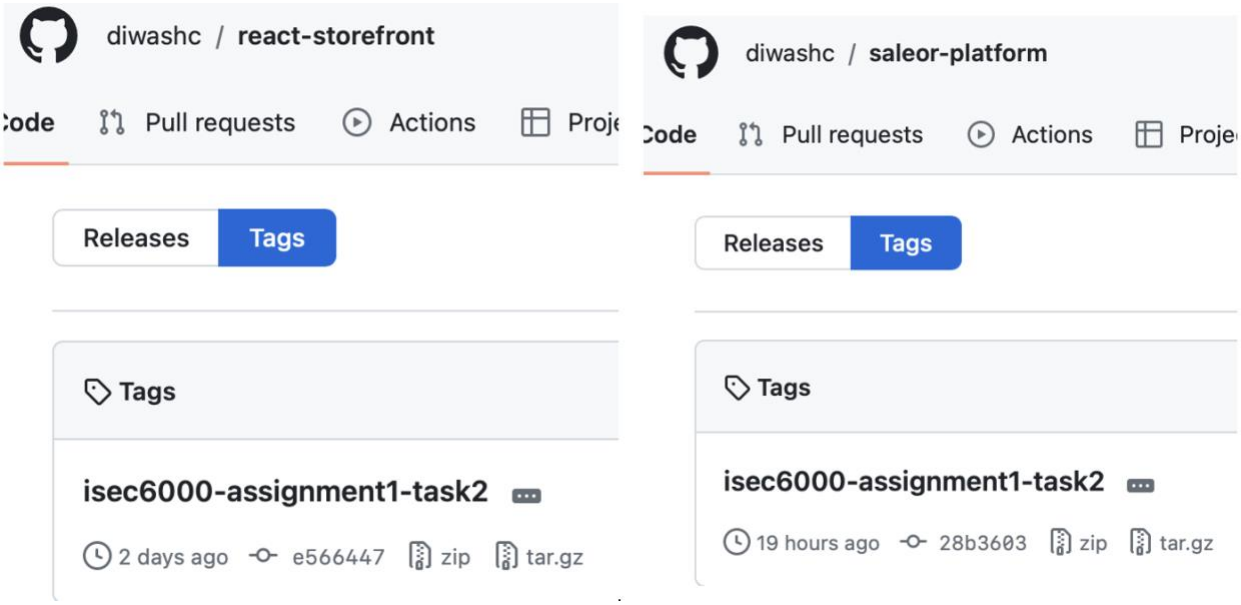

Figure 13: Tag for Task 2

The tags for respective repositories are committed and pushed and tags are created as shown in figure 12 and 13.

4. Task 3: Implementing Security Measures

4.1. Ensuring Secure Configuration of containers

To protect containerized environments, container security is used. It involves the maintenance and implementation of security controls that protects underlying infrastructures and containers. (Tigera, 2023) To ensure secure configuration of containers the following steps are used in this project.

- Continuously scanning image for any vulnerabilities
- Removing unnecessary service or package from container images
- Avoid running containers as root
- Use only trusted images
- Set resource constraints such as CPU and memory limits to prevent containers from excessive use of resources.

4.2. Implementation of container image vulnerability

Among open-source security scanner, trivy is the most popular as it is more reliable, fast, and easy to use. (Trivy, 2023) For using trivy there are different ways to install it but here it is installed within a docker container using "docker run aquasec/trivy" command".

Now to run trivy and check it for saleor dashboard the following command was used.

docker run --rm -v /var/run/docker.sock:/var/run/docker.sock aquasec/trivy image ghcr.io/saleor/saleor-dashboard:3.15.2

Figure 14: Image Scan for Saleor Dashboard

Figure 14 displays that there are no vulnerabilities for saleor dashboard in the trivy scan result. Now for image vulnerability scanning for react storefront the following command was used.

docker run --rm -v /var/run/docker.sock:/var/run/docker.sock aquasec/trivy image react-storefront-storefront:latest

Figure 15:Image Scan for React Storefront

Figure 15 displays the vulnerabilities that were found while scanning react storefront container. And figure 16 and 17 further displays the detailed result which displays the library, vulnerability, severity, status, installed version, fixed version and title.

Library	Vulnerability	Severity	Status	Installed Version	Fixed Version	Title
apt	CVE-2011-3374	LOW	affected	1.8.2.3		It was found that apt-key in apt, all versions, do not correctly... https://avd.aquasec.com/nvd/cve-2011-3374
bash	CVE-2019-18276			5.0-4		bash: when effective UID is not equal to its real UID the... https://avd.aquasec.com/nvd/cve-2019-18276
bsdutils	CVE-2021-37600			1:2.33.1-0.1		util-linux: integer overflow can lead to buffer overflow in get_sem_elements() in sys-utils/ipcutils.c... https://avd.aquasec.com/nvd/cve-2021-37600
	CVE-2022-0563					partial disclosure of arbitrary files in chfn and chsh when compiled with... https://avd.aquasec.com/nvd/cve-2022-0563
coreutils	CVE-2016-2781		will_not_fix	8.30-3		coreutils: Non-privileged session can escape to the parent session in chroot https://avd.aquasec.com/nvd/cve-2016-2781
	CVE-2017-18018		affected			coreutils: race condition vulnerability in chown and chgrp https://avd.aquasec.com/nvd/cve-2017-18018
e2fsprogs	CVE-2022-1304	HIGH		1.44.5-1+deb10u3		out-of-bounds read/write via crafted filesystem https://avd.aquasec.com/nvd/cve-2022-1304
fdisk	CVE-2021-37600	LOW		2.33.1-0.1		util-linux: integer overflow can lead to buffer overflow in get_sem_elements() in sys-utils/ipcutils.c... https://avd.aquasec.com/nvd/cve-2021-37600
	CVE-2022-0563					partial disclosure of arbitrary files in chfn and chsh when compiled with... https://avd.aquasec.com/nvd/cve-2022-0563
gcc-8-base	CVE-2018-12886	HIGH	will_not_fix	8.3.0-6		gcc: spilling of stack protection address in cfgexpand.c and function.c leads to... https://avd.aquasec.com/nvd/cve-2018-12886
	CVE-2019-15847					gcc: POWER9 "DARN" RNG intrinsic produces repeated output https://avd.aquasec.com/nvd/cve-2019-15847
gpgv	CVE-2019-14855	LOW		2.2.12-1+deb10u2		gnupg2: OpenPGP Key Certification Forgeries with SHA-1 https://avd.aquasec.com/nvd/cve-2019-14855
	CVE-2022-3219		affected			denial of service issue (resource consumption) using compressed packets https://avd.aquasec.com/nvd/cve-2022-3219
libapt-pkg5.0	CVE-2011-3374			1.8.2.3		It was found that apt-key in apt, all versions, do not correctly... https://avd.aquasec.com/nvd/cve-2011-3374

Figure 16: Vulnerability in react storefront

libcom-err2	CVE-2022-1304	HIGH		1.44.5-1+deb10u3		out-of-bounds read/write via crafted filesystem https://avd.aquasec.com/nvd/cve-2022-1304
libdb5.3	CVE-2019-8457	CRITICAL	will_not_fix	5.3.28+dfsg1-0.5		heap out-of-bound read in function rtreenode() https://avd.aquasec.com/nvd/cve-2019-8457
libexpat1	CVE-2013-0340	LOW	affected	2.2.6-2+deb10u6		expat: internal entity expansion https://avd.aquasec.com/nvd/cve-2013-0340
libext2fs2	CVE-2022-1304	HIGH		1.44.5-1+deb10u3		out-of-bounds read/write via crafted filesystem https://avd.aquasec.com/nvd/cve-2022-1304
libfdisk1	CVE-2021-37600	LOW		2.33.1-0.1		util-linux: integer overflow can lead to buffer overflow in get_sem_elements() in sys-utils/ipcutils.c... https://avd.aquasec.com/nvd/cve-2021-37600
	CVE-2022-0563					partial disclosure of arbitrary files in chfn and chsh when compiled with... https://avd.aquasec.com/nvd/cve-2022-0563
libfreetype6	CVE-2022-31782			2.9.1-3+deb10u3		ftbench.c in FreeType Demo Programs through 2.12.1 has a heap-based bu https://avd.aquasec.com/nvd/cve-2022-31782
libgcc1	CVE-2018-12886	HIGH	will_not_fix	1:8.3.0-6		gcc: spilling of stack protection address in cfgexpand.c and function.c leads to... https://avd.aquasec.com/nvd/cve-2018-12886
	CVE-2019-15847					gcc: POWER9 "DARN" RNG intrinsic produces repeated output https://avd.aquasec.com/nvd/cve-2019-15847
libgcrypt20	CVE-2021-33560		affected	1.8.4-5+deb10u1		libgcrypt: mishandles ElGamal encryption because it lacks exponent blinding to address a... https://avd.aquasec.com/nvd/cve-2021-33560
	CVE-2019-13627	MEDIUM				ECDSA timing attack allowing private key leak https://avd.aquasec.com/nvd/cve-2019-13627
	CVE-2018-6829	LOW				libgcrypt: ElGamal implementation doesn't have semantic security due to incorrectly encoded plaintexts... https://avd.aquasec.com/nvd/cve-2018-6829

Figure 17: Vulnerability in react storefront 2

5. Task 4: Architecture Visualization

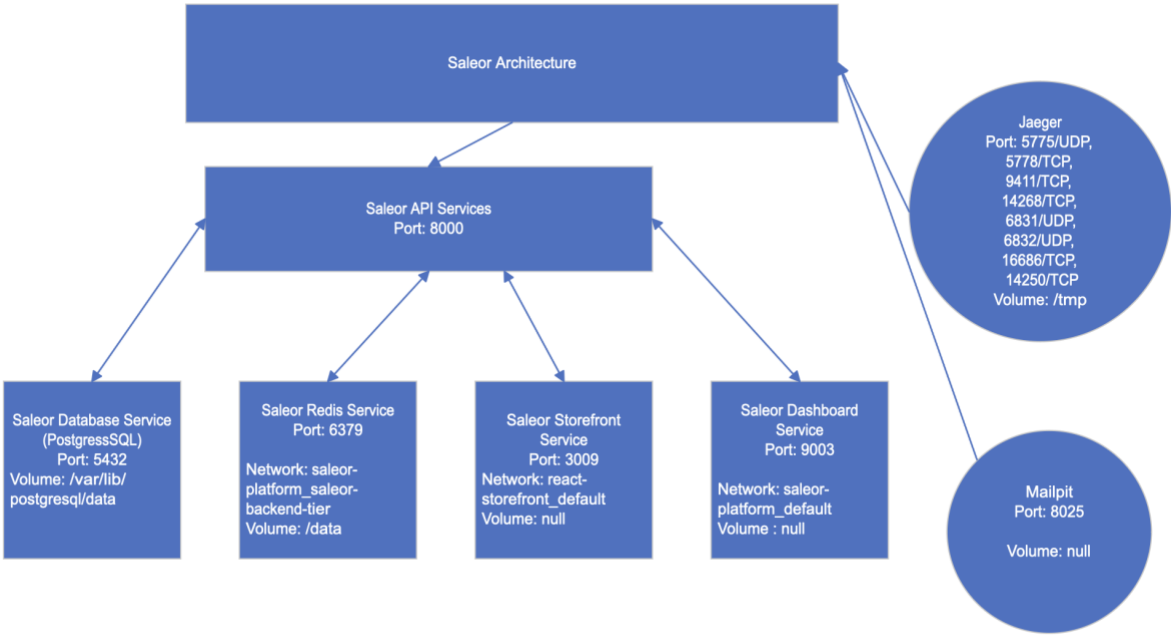

Figure 18: Architecture Diagram for Saleor service

All Saleor services interact with API services to interact with each other. All the services operate in their respective ports so that there is no clash between their operations.

6. Conclusion

The cluster was successfully created and deployed in google cloud with location and node pool configuration. With the help of git repositories saleor components were configured and build. Changes to the ports were made to change the URL in which it was running previously. Implementation of security measures were done to secure the containers from vulnerabilities. The architecture diagram that encapsulates the main functionalities of each Saleor service was created.

Reference

Betterexplained. (2023, 05 20). *Betterexplained*. Retrieved from Betterexplained: https://betterexplained.com/articles/a-visual-guide-to-version-control/

Saleor. (2023, 09 09). *saleor*. Retrieved from saleor: https://learn.saleor.io/intro/architecture-overview/

Trivy. (2023, 09 08). *Trivy*. Retrieved from Trivy: https://trivy.dev

Tigera. (2023, 09 02). *Tigera*. Retrieved from Tigera: https://www.tigera.io/learn/guides/container-security-best-practices/