International Institute of Informational Technology, Bangalore

Serverless platform - MOSIP Resident Services

Kernel Service functions

MT2021092
PIYUSH KUMAR

MT2021062
PRUDHVI NAG K.

MT2021030
BHAVYA VIJAYVERGIYA

MT2021109
RISHABH TALWAR

MT2021110
ROOPAM PATIL

UNDER :
Prof B Thangaraju

Guided by:
Sanath Varambally

Tasks Assigned and Performed :

1. **Technologies and Tools Learned and Used :**
    a. Knative
    b. Kubernetes
    c. Docker
    d. Kind
    e. Serverless Architecture

2. **Installing KNative :**
    Reference Link Provided : https://knative.dev/docs/install/
    Install Knative using quickstart :

        a. kind (Kubernetes in Docker) or minikube to enable you to run a local Kubernetes cluster with Docker container nodes.You can install the Serving component, Eventing component, or both on your cluster by using one of the following deployment options:

- ✓ Use the **Knative Quickstart plugin** to install a preconfigured, local distribution of Knative for development purposes.
- ✓ Use a YAML-based installation to install a production ready deployment:
    - **Install Knative Serving by using YAML**
    - **Install Knative Eventing by using YAML**
- ✓ Use the **Knative Operator** to install and configure a production ready deployment.
- ✓ Follow the documentation for vendor managed **Knative offerings**.

```
curl -Lo ./kind https://kind.sigs.k8s.io/dl/v0.12.0/kind-linux-amd64
chmod +x ./kind
mv ./kind /some-dir-in-your-PATH/kind
```

b. The Kubernetes CLI (kubectl) to run commands against Kubernetes clusters. You can use kubectl to deploy applications, inspect and manage cluster resources, and view logs.

```
Download the latest release with the command:

curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"

Validate the binary (optional)

curl -LO "https://dl.k8s.io/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256"
echo "$(cat kubectl.sha256)  kubectl" | sha256sum --check

Install kubectl

sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

Test to ensure the version you installed is up-to-date:

kubectl version --client
```

c. Install the Knative CLI :

i. Download the binary for your system from the quickstart release page.

ii. Rename the file to remove the OS and architecture information. For example, rename kn-quickstart-amd64 to kn-quickstart.

iii. Make the plugin executable. For example, chmod +x kn-quickstart.

iv. Move the executable binary file to a directory on your PATH, for example, in /usr/local/bin.

v. Verify that the plugin is working by running the command: kn quickstart --help

d. Run the Knative quickstart plugin :
The quickstart plugin completes the following functions:

1. **Checks if you have the selected Kubernetes instance installed**
2. **Creates a cluster called knative**
3. **Installs Knative Serving** with Kourier as the default networking layer, and sslip.io as the DNS
4. **Installs Knative Eventing** and creates an in-memory Broker and Channel implementation

```
kubectl cluster-info --context kind-knative

Thanks for using kind! 😊

Installing Knative Serving v1.2.0 ...
    CRDs installed...
    Core installed...
    Finished installing Knative Serving
Installing Kourier networking layer v1.2.0 ...
    Kourier installed...
    Ingress patched...
    Finished installing Kourier Networking layer
Configuring Kourier for Kind...
    Kourier service installed...
    Domain DNS set up...
    Finished configuring Kourier
Installing Knative Eventing v1.2.0 ...
    CRDs installed...
    Core installed...
    In-memory channel installed...
    Mt-channel broker installed...
    Example broker installed...
    Finished installing Knative Eventing
Knative install took: 4m34s
Now have some fun with Serverless and Event Driven Apps!
```

```
prudhvi@prudhvi-IdeaPad-3-15ITL6:~$ sudo kn quickstart kind
[sudo] password for prudhvi:
Running Knative Quickstart using Kind
✅ Checking dependencies...
    Kind version is: 0.11.1

Creating Kind cluster...
Creating cluster "knative" ...
 ✓ Ensuring node image (kindest/node:v1.22.4) 🖼
 ✓ Preparing nodes 📦
 ✓ Writing configuration 📜
 ✓ Starting control-plane 🕹️
 ✓ Installing CNI 🔌
 ✓ Installing StorageClass 💾
 ✓ Waiting ≤ 2m0s for control-plane = Ready ⏳
 • Ready after 19s 💚
Set kubectl context to "kind-knative"
You can now use your cluster with:

kubectl cluster-info --context kind-knative

Thanks for using kind! 😊

Installing Knative Serving v1.2.0 ...
    CRDs installed...
    Core installed...
    Finished installing Knative Serving
Installing Kourier networking layer v1.2.0 ...
    Kourier installed...
    Ingress patched...
    Finished installing Kourier Networking layer
Configuring Kourier for Kind...
    Kourier service installed...
    Domain DNS set up...
    Finished configuring Kourier
Installing Knative Eventing v1.2.0 ...
    CRDs installed...
    Core installed...
    In-memory channel installed...
    Mt-channel broker installed...
    Example broker installed...
    Finished installing Knative Eventing
```

4. Running a Sample Application (Hello World) :

a. Deploying the hello world service

```
Deploy the Service by running the command:

sudo kn service create hello --image
gcr.io/knative-samples/helloworld-go --port 8080 --env TARGET=World
```

```
swaraj@swaraj-Inspiron-5567:~$ sudo kn service create hello --image gcr.io/knative-samples/helloworld-go --port 8080 --env TARGET=World
[sudo] password for swaraj:
Creating service 'hello' in namespace 'default':

  2.187s The Route is still working to reflect the latest desired specification.
  2.362s ...
  2.756s Configuration "hello" is waiting for a Revision to become ready.
129.827s ...
130.654s Ingress has not yet been reconciled.
132.475s Waiting for load balancer to be ready
133.759s Ready to serve.

Service 'hello' created to latest revision 'hello-00001' is available at URL:
http://hello.default.127.0.0.1.sslip.io
```

b. View all the services

```
View a list of Knative services by running the command:

kn service list
```

```
prudhvi@prudhvi-IdeaPad-3-15ITL6:~$ sudo kn service list
No services found.
prudhvi@prudhvi-IdeaPad-3-15ITL6:~$ sudo kn service create hello --image gcr.io/knative-samples/helloworld-go --port 8080 --env TARGET=World
Creating service 'hello' in namespace 'default':

  0.034s The Route is still working to reflect the latest desired specification.
  0.056s ...
  0.073s Configuration "hello" is waiting for a Revision to become ready.
 35.685s ...
 35.709s Ingress has not yet been reconciled.
 35.746s Waiting for load balancer to be ready
 35.930s Ready to serve.

Service 'hello' created to latest revision 'hello-00001' is available at URL:
http://hello.default.127.0.0.1.sslip.io
```

```
prudhvi@prudhvi-IdeaPad-3-15ITL6:~$ sudo kn service list
NAME    URL                                   LATEST       AGE    CONDITIONS   READY   REASON
hello   http://hello.default.127.0.0.1.sslip.io   hello-00001   2m46s   3 OK / 3     True
```

c. Access your Knative Service by opening the previous URL in your browser or by running the command:

```
echo "Accessing URL $(kn service describe hello -o url)"
```

```
prudhvi@prudhvi-IdeaPad-3-15ITL6:~$ sudo echo "Accessing URL $(kn service describe hello -o url)"
Error: Get "https://127.0.0.1:46271/apis/serving.knative.dev/v1/namespaces/default/services/hello": dial tcp 127.0.0.1:46271: connect: connection refused
Run 'kn --help' for usage
Accessing URL
```

```
curl "$(kn service describe hello -o url)"
```


```
Hello World!
```

5. Install Kubernetes Dashboard using Kubectl :

a. Deploy the Kubernetes dashboard using Kubectl:

```
sudo kubectl apply -f
https://raw.githubusercontent.com/kubernetes/dashboard/v2.3.1/
aio/deploy/recommended.yaml
```

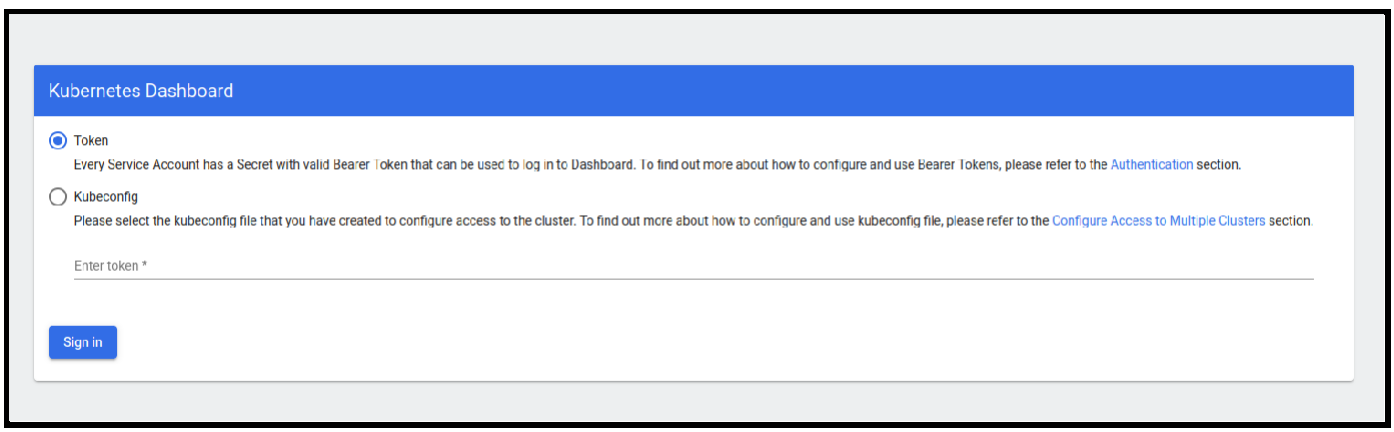
```
prudhvi@prudhvi-IdeaPad-3-15ITL6:~$ sudo kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.3.1/aio/deploy/recommended.yaml
namespace/kubernetes-dashboard created
serviceaccount/kubernetes-dashboard created
service/kubernetes-dashboard created
secret/kubernetes-dashboard-certs created
secret/kubernetes-dashboard-csrf created
secret/kubernetes-dashboard-key-holder created
configmap/kubernetes-dashboard-settings created
role.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrole.rbac.authorization.k8s.io/kubernetes-dashboard created
rolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
deployment.apps/kubernetes-dashboard created
service/dashboard-metrics-scraper created
Warning: spec.template.metadata.annotations[seccomp.security.alpha.kubernetes.io/pod]: deprecated since v1.19; use the "seccompProfile" field instead
deployment.apps/dashboard-metrics-scraper created
```

b. Access Kubernetes Dashboard using Kubectl :

```
sudo kubectl proxy
```

```
prudhvi@prudhvi-IdeaPad-3-15ITL6:~$ sudo kubectl proxy
\Starting to serve on 127.0.0.1:8001
```

c. Access Kubernetes Dashboard using following url :
http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/



d. Kubernetes Dashboard Authentication :

i. Create a Service Account :

```
sudo kubectl create serviceaccount dashboard-admin-sa
```

```
prudhvi@prudhvi-IdeaPad-3-15ITL6:~$ sudo kubectl create serviceaccount dashboard
-admin-sa
[sudo] password for prudhvi:
serviceaccount/dashboard-admin-sa created
prudhvi@prudhvi-IdeaPad-3-15ITL6:~$
```

ii. Bind the dashboard-admin-service-account service account to the cluster-admin role and get secrets

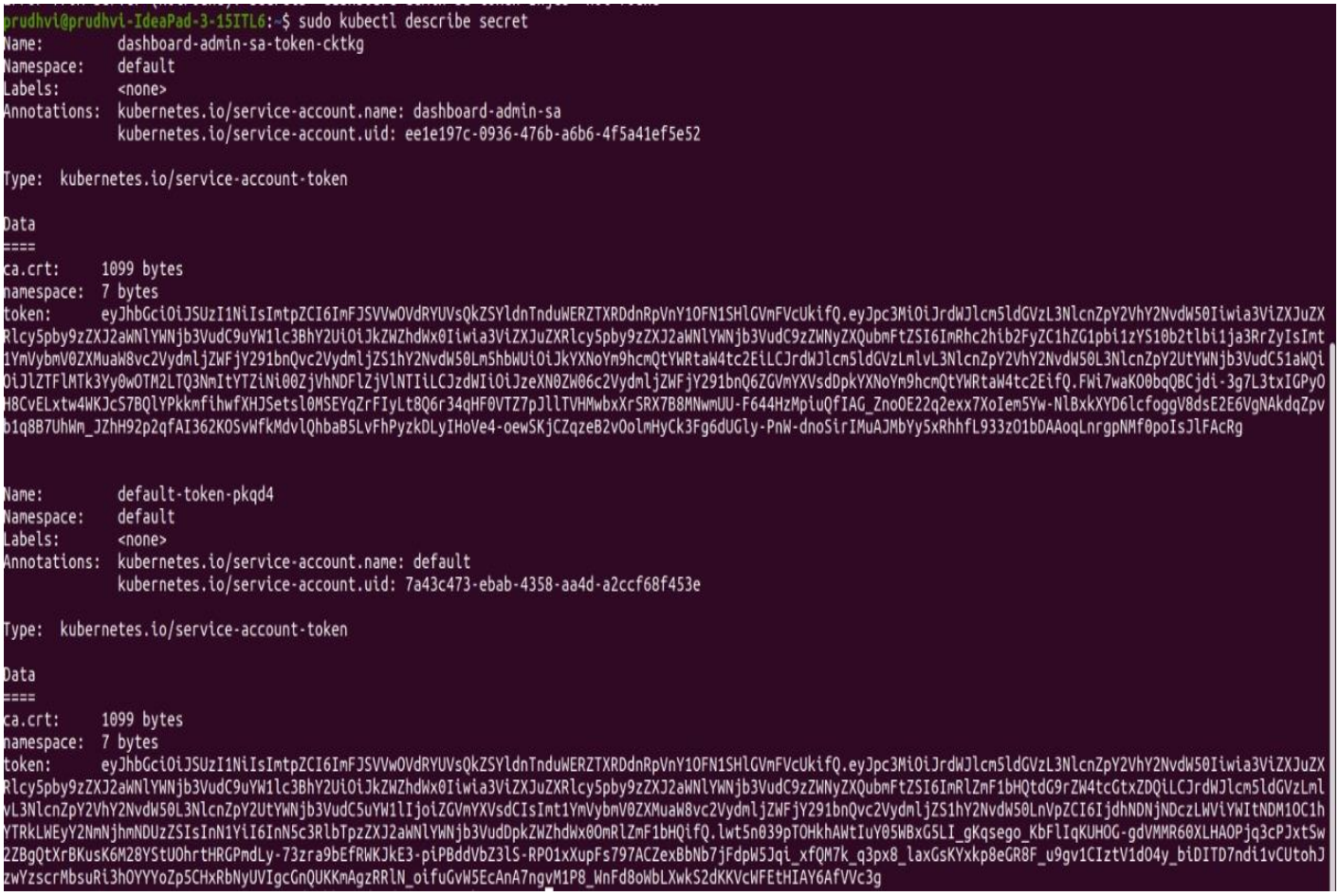
```
sudo kubectl create clusterrolebinding dashboard-admin-sa
--clusterrole=cluster-admin
--serviceaccount=default:dashboard-admin-sa

kubectl get secrets
```

```
prudhvi@prudhvi-IdeaPad-3-15ITL6:~$ sudo kubectl create clusterrolebinding dashb
oard-admin-sa1 --clusterrole=cluster-admin --serviceaccount=default:dashboard-ad
min-sa
clusterrolebinding.rbac.authorization.k8s.io/dashboard-admin-sa1 created
```

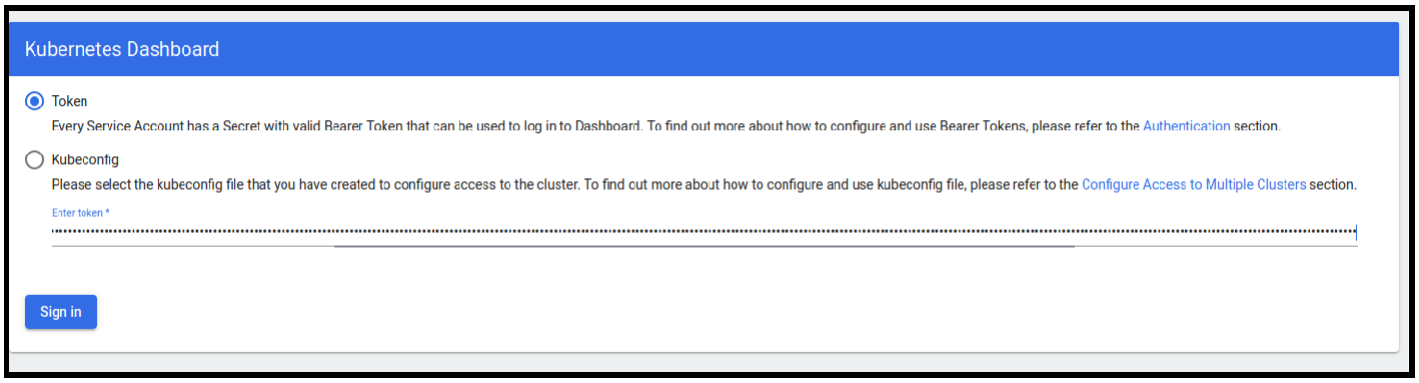
```
prudhvi@prudhvi-IdeaPad-3-15ITL6:~$ sudo kubectl get secrets
NAME                                 TYPE                                  DATA   AGE
dashboard-admin-sa-token-cktkg       kubernetes.io/service-account-token   3      4m4s
default-token-pkqd4                  kubernetes.io/service-account-token   3      30m
```

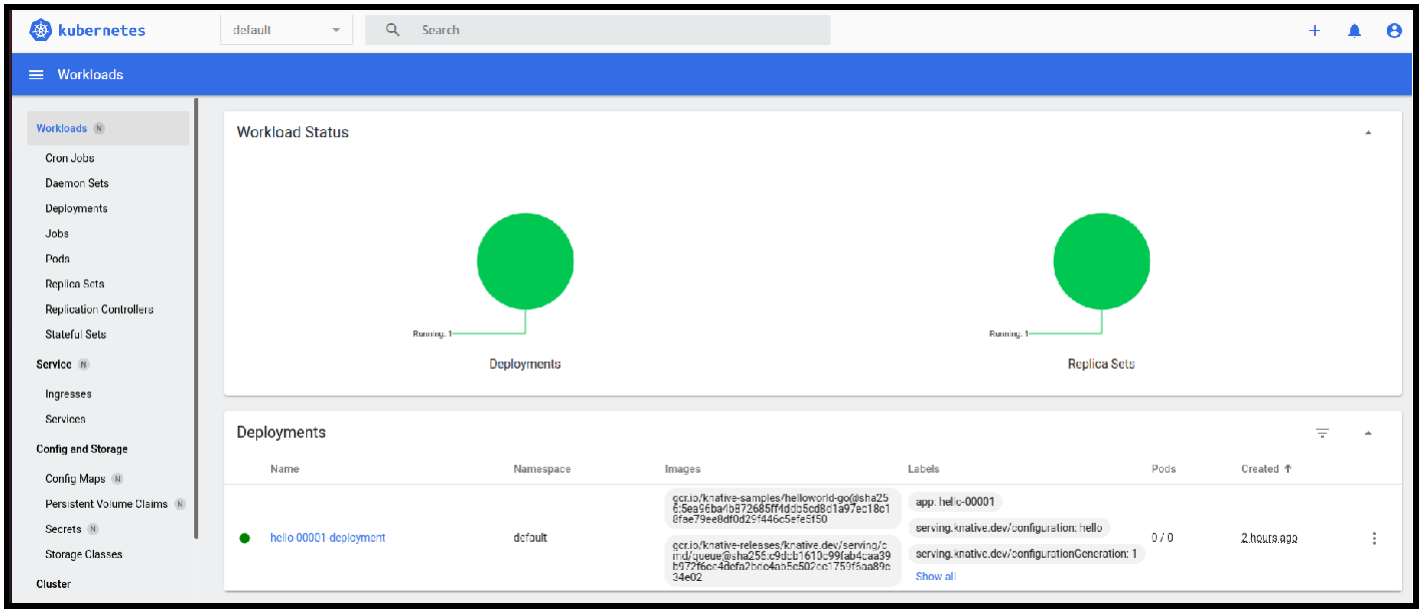
iii. Use kubectl describe to get the access token:

```
sudo kubectl describe secret
dashboard-admin-sa-token-lnjc8
```



iv. Enter the token in the login field :

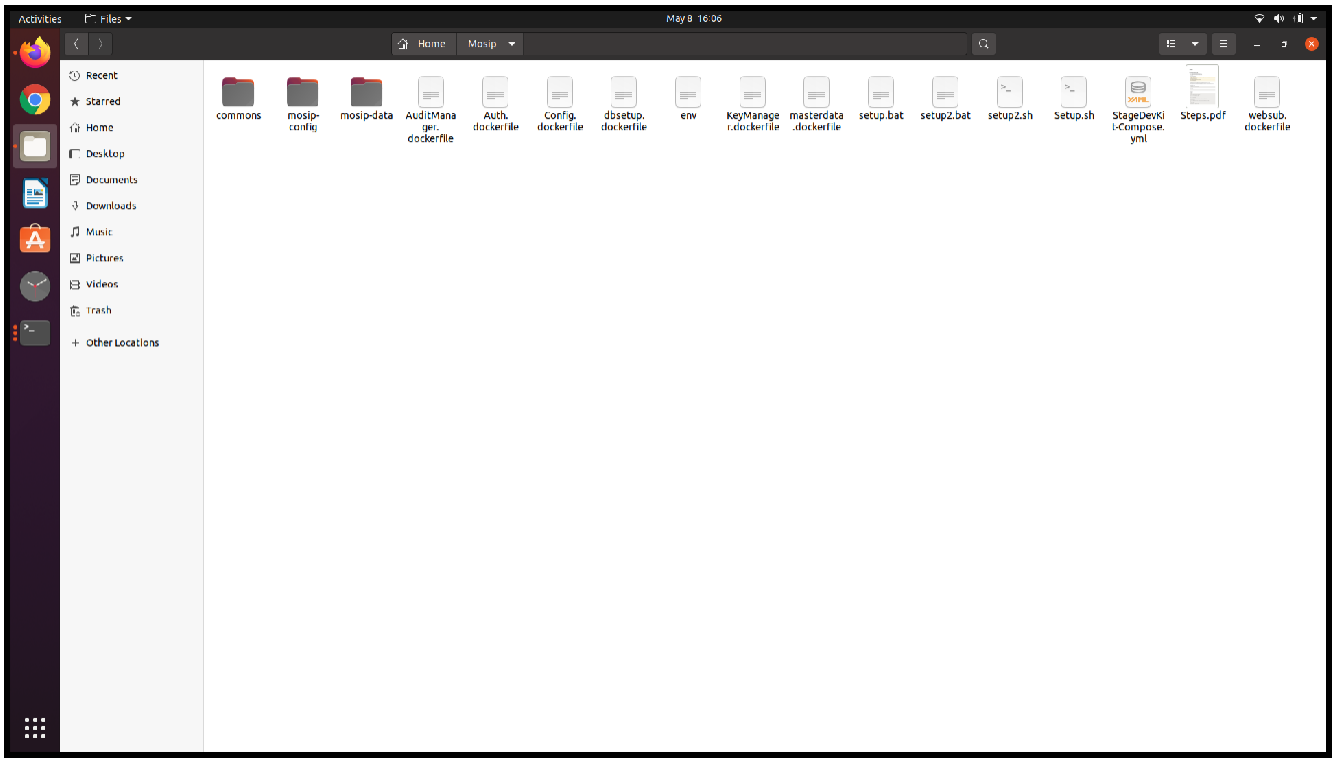




6. Kernel Service Setup :

```
Git repo clone (branch: 1.1.5.4)

1. git clone https://github.com/mosip/commons.git : DDL
git clone https://github.com/mosip/mosip-data.git : DML

2. cd commons

3. mvn clean install -DskipTests
Changed mosip.auth.adapter.impl.basepackage to
io.mosip.kernel.auth.defaultadapter in
AuditManagerBootApplication
```

```
Added spring.h2.console.settings.web-allow-others=true in
commons/kernel/kernel-auditmanager-
service/target/classes/application-local.properties

mvn clean install -DskipTests

4. java -jar
-Dloader.path=kernel\kernel-auth-adapter\target\kernel-auth-adapter-
1.2.0-rc1.jar -jar -
Dspring.profiles.active=local
kernel\kernel-auth-service\target\kernel-auth-service-1.2.0-rc1.jar

5. wget
https://repo1.maven.org/maven2/io/mosip/kernel/kernel-config-server/
1.1.2/kernel-config-server-1.1.2.jar -P
mosip-config

6. docker volume create --name=mosip_config

7. docker-compose -f StageDevKit-Compose.yml up
```



StageDevKit-Compose :

```
version: '3.8'
volumes:
   shared-workspace:
       name: "devkit-distributed-file-system"
       driver: local
   mosip_config:
       external: true
       name: mosip_config
services:
   mosip_auth_service:
       build:
           context: .
           dockerfile: Auth.dockerfile
       image: kernel/auth:v1
       container_name: mosip_auth_service
       ports:
           - 8091:8091
       volumes:
           - shared-workspace:/opt/workspace
   mosip_config_service:
       build:
           context: .
           dockerfile: Config.dockerfile
       image: kernel/config:v1
       container_name: mosip_config_service
       ports:
           - 51000:51000
       volumes:
           - shared-workspace:/opt/workspace
           - mosip_config:/config
       environment:
           - AUTH_SERVICE=http://mosip_auth_service:8091
       depends_on:
           - mosip_auth_service
   mosip_audit_service:
       build:
           context: .
           dockerfile: AuditManager.dockerfile
```

```
        image: kernel/auditmanager:v1
        container_name: mosip_audit_service
        ports:
            - 8081:8081
        volumes:
            - shared-workspace:/opt/workspace
        environment:
            - AUTH_SERVICE=http://mosip_auth_service:8091
            - CONFIG_SERVICE=mosip_config_service:51000
        depends_on:
            - mosip_auth_service
    postgres:
        image: debezium/postgres
        container_name: postgres
        ports:
            - 5432:5432
        volumes:
            - shared-workspace:/opt/workspace
        environment:
            - POSTGRES_PASSWORD=root
            - PGDATA=/data/pgdata
            - POSTGRES_DB=kernel111
```

Setup.sh

```
#!/bin/sh
repository="https://github.com/mosip/commons.git"
local="/home/nupur/Desktop/IIITB/Semester/3rdSem/mosip/commons"
git clone -b 1.1.5.4 "$repository" "$local"
cd commons
mvn clean install -DskipTests
loader_path="kernel\kernel-auth-adapter\target\kernel-auth-adapter-1.2.0-rc1.jar"
path="kernel\kernel-auth-service\target\kernel-auth-service-1.2.0-rc1.jar"
spring_profile="local"
java -jar -Dloader.path="$loader_path" -jar
-Dspring.profiles.active="$spring_profile" "$path"
cd ..
# get the config jar
```

```
wget
https://repo1.maven.org/maven2/io/mosip/kernel/kernel-config-server/1.1.2/
kernel-config-server-1.1.2.jar -P mosip-config
docker volume create --name=mosip_config
# build docker images
docker-compose -f StageDevKit-Compose.yml up
```

Setup2.sh

```
VERSION=1.2.0-rc1
#git clone https://github.com/mosip/commons.git
#cd commons


#git checkout $VERSION

#mvn clean install -DskipTests

#java -jar
-Dloader.path=kernel/kernel-auth-adapter/target/kernel-auth-adapter-1.2.0-
rc1.jar -jar -Dspring.profiles.active=local
kernel/kernel-auth-service/target/kernel-auth-service-1.2.0-rc1.jar


#cd ..

#wget
https://repo1.maven.org/maven2/io/mosip/kernel/kernel-config-server/1.1.2/
kernel-config-server-1.1.2.jar -P mosip-config

echo $VERSION
docker build -f Auth.dockerfile --build-arg version=$VERSION -t
kernel/auth:v1 .
docker build -f AuditManager.dockerfile --build-arg version=$VERSION -t
kernel/auditmanager:v1 .
docker build -f Config.dockerfile --build-arg version=$VERSION -t
kernel/config:v1 .
```

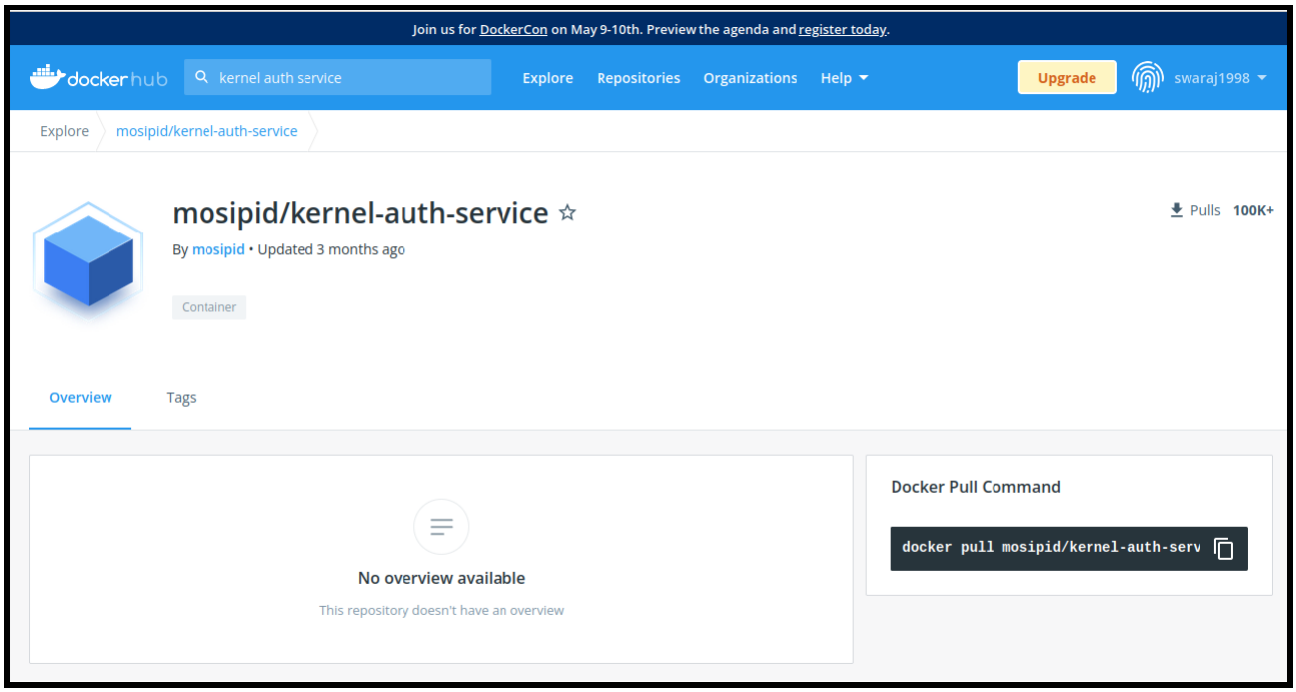
```
#docker build -f test.dockerfile
docker volume create --name=mosip_config

docker rm -f $(docker ps -a -q)

docker-compose -f StageDevKit-Compose.yml -d up
```

Following output will be generated
After running command :tree –L 1

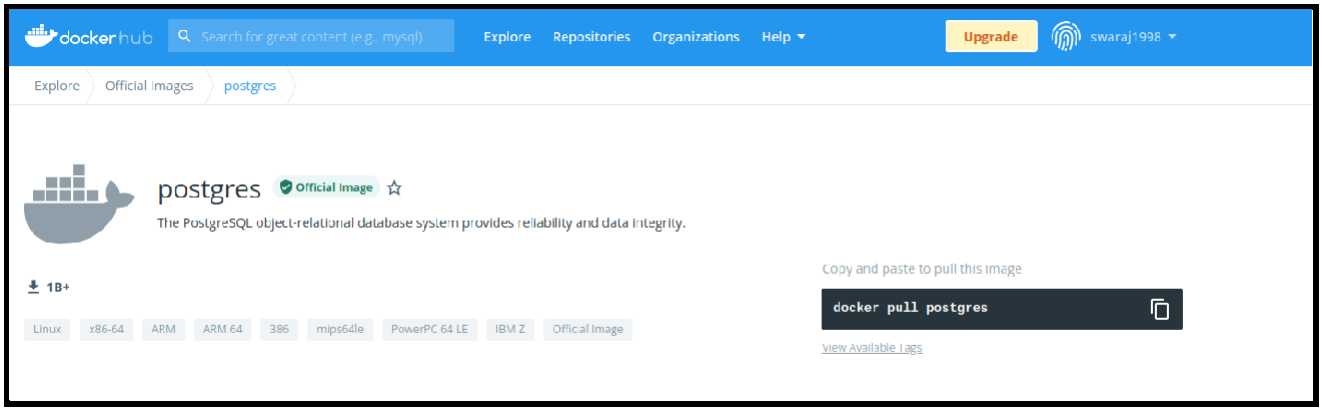
```
.
├── AuditManager.dockerfile
├── Auth.dockerfile
├── commons
├── Config.dockerfile
├── dbsetup.dockerfile
├── env
├── KeyManager.dockerfile
├── masterdata.dockerfile
├── mosip-config
├── mosip-data
├── setup2.bat
├── setup2.sh
├── setup.bat
├── Setup.sh
├── StageDevKit-Compose.yml
├── Steps.pdf
└── websub.dockerfile

3 directories, 14 files
```

7. **Running Kernel Auth Service :**

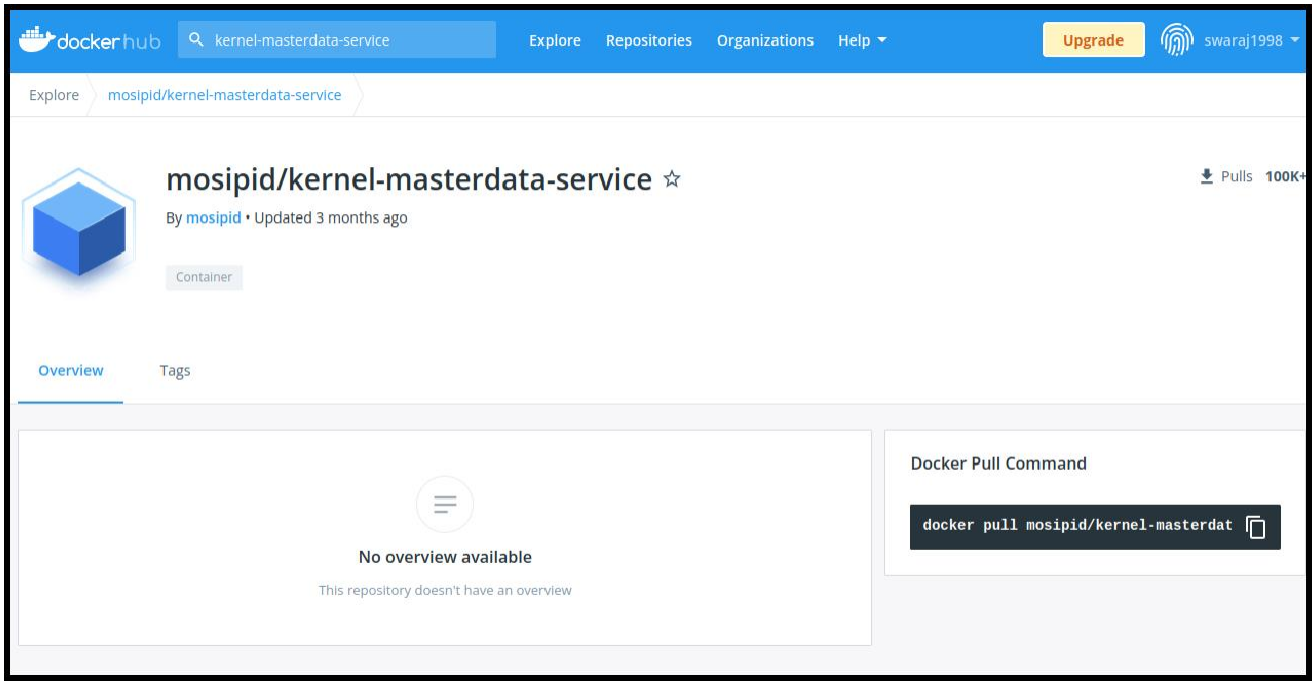
    a. Pulling and running ‘kernel-auth-service’ using KNative Serving.



b. Pulling and running postgres:



c. Pulling and running ‘kernel-masterdata-service’ using KNative Serving.



Yaml File

```
---
kind: Pod
metadata:
 labels:
   app: demo.40
 namespace: default
spec:
 containers:
 - image: mosipid/kernel-auth-service
   name: kernel-auth-service
   ports:
   - containerPort: 8091
     name: portname.0
     protocol: tcp
   volumeMounts:
   - mountPath: /opt/workspace
     name: pvo.0
 terminationGracePeriodSeconds: 0
 volumes:
 - name: pvo.0
   persistentVolumeClaim:
```

```
    claimName: claimname.0

---
kind: Pod
metadata:
 labels:
   app: demo.91
 namespace: default
spec:
 containers:
 - image: mosipid/kernel-masterdata-service
   name: kernel-masterdata-service
   ports:
   - containerPort: 8092
     name: portname.0
     protocol: tcp
   volumeMounts:
   - mountPath: /opt/workspace
     name: pvo.0
 terminationGracePeriodSeconds: 0
 volumes:
 - name: pvo.0
   persistentVolumeClaim:
     claimName: claimname.0
```

Docker Compose File:

```
version: '3.8'
volumes:
   shared-workspace:
       name: "devkit-distributed-file-system"
       driver: local
   mosip_config:
       external: true
       name: mosip_config
services:
   kernel-auth-service:
       image: mosipid/kernel-auth-service
       container_name: kernel-auth-service
```

```
    ports:
      - 8091:8091
    volumes:
      - shared-workspace:/opt/workspace

postgres:
    image: debezium/postgres
    container_name: postgres
    ports:
      - 5432:5432
    volumes:
      - shared-workspace:/opt/workspace
    environment:
      - POSTGRES_PASSWORD=root
      - PGDATA=/data/pgdata
      - POSTGRES_DB=kernel111

kernel-masterdata-service:
    image: mosipid/kernel-masterdata-service
    container_name: kernel-masterdata-service
    ports:
      - 8092:8092
    volumes:
      - shared-workspace:/opt/workspace
```

prudhvi@prudhvi-IdeaPad-3-15ITL6:docker-compose pull

```
Pulling kernel-auth-service        ... done
Pulling postgres                   ... done
Pulling kernel-masterdata-service  ... done
```

```
Attaching to kernel-masterdata-service, postgres, kernel-auth-service
kernel-auth-service          | Exception in thread "main" java.util.zip.ZipException: zip file is empty
kernel-auth-service          |  at java.base/java.util.zip.ZipFile$Source.zerror(ZipFile.java:1607)
kernel-auth-service          |  at java.base/java.util.zip.ZipFile$Source.findEND(ZipFile.java:1410)
kernel-auth-service          |  at java.base/java.util.zip.ZipFile$Source.initCEN(ZipFile.java:1504)
kernel-auth-service          |  at java.base/java.util.zip.ZipFile$Source.<init>(ZipFile.java:1308)
kernel-auth-service          |  at java.base/java.util.zip.ZipFile$Source.get(ZipFile.java:1271)
kernel-auth-service          |  at java.base/java.util.zip.ZipFile$CleanableResource.<init>(ZipFile.java:831)
kernel-auth-service          |  at java.base/java.util.zip.ZipFile$CleanableResource$FinalizableResource.<init>(ZipFile.java:857)
kernel-auth-service          |  at java.base/java.util.zip.ZipFile$CleanableResource.get(ZipFile.java:846)
kernel-auth-service          |  at java.base/java.util.zip.ZipFile.<init>(ZipFile.java:248)
kernel-auth-service          |  at java.base/java.util.zip.ZipFile.<init>(ZipFile.java:177)
kernel-auth-service          |  at java.base/java.util.jar.JarFile.<init>(JarFile.java:350)
kernel-auth-service          |  at java.base/java.util.jar.JarFile.<init>(JarFile.java:321)
kernel-auth-service          |  at java.base/java.util.jar.JarFile.<init>(JarFile.java:287)
kernel-auth-service          |  at org.springframework.boot.loader.jar.JarFile.<init>(JarFile.java:115)
kernel-auth-service          |  at org.springframework.boot.loader.jar.JarFile.<init>(JarFile.java:109)
kernel-auth-service          |  at org.springframework.boot.loader.jar.JarFile.<init>(JarFile.java:95)
kernel-auth-service          |  at org.springframework.boot.loader.jar.JarFile.<init>(JarFile.java:86)
kernel-auth-service          |  at org.springframework.boot.loader.archive.JarFileArchive.<init>(JarFileArchive.java:60)
kernel-auth-service          |  at org.springframework.boot.loader.archive.JarFileArchive.<init>(JarFileArchive.java:56)
kernel-auth-service          |  at org.springframework.boot.loader.archive.ExplodedArchive.getNestedArchive(ExplodedArchive.java:119)
kernel-auth-service          |  at org.springframework.boot.loader.archive.ExplodedArchive.getNestedArchives(ExplodedArchive.java:106)
kernel-auth-service          |  at org.springframework.boot.loader.PropertiesLauncher.getClassPathArchives(PropertiesLauncher.java:449)
kernel-auth-service          |  at org.springframework.boot.loader.Launcher.launch(Launcher.java:49)
kernel-auth-service          |  at org.springframework.boot.loader.PropertiesLauncher.main(PropertiesLauncher.java:593)
```

```
postgres                     | syncing data to disk ... ok
postgres                     |
postgres                     | Success. You can now start the database server using:
postgres                     |
postgres                     |     pg_ctl -D /data/pgdata -l logfile start
postgres                     |
postgres                     |
postgres                     | WARNING: enabling "trust" authentication for local connections
postgres                     | You can change this by editing pg_hba.conf or using the option -A, or
postgres                     | --auth-local and --auth-host, the next time you run initdb.
postgres                     | waiting for server to start....LOG:  database system was shut down at 2022-05-08 11:23:21 GMT
postgres                     | LOG:  MultiXact member wraparound protections are now enabled
postgres                     | LOG:  autovacuum launcher started
postgres                     | LOG:  database system is ready to accept connections
postgres                     |  done
postgres                     | server started
postgres                     | CREATE DATABASE
postgres                     |
postgres                     |
postgres                     | /usr/local/bin/docker-entrypoint.sh: sourcing /docker-entrypoint-initdb.d/init-permissions.sh
postgres                     |
postgres                     | LOG:  received fast shutdown request
postgres                     | waiting for server to shut down...LOG:  aborting any active transactions
postgres                     | LOG:  autovacuum launcher shutting down
postgres                     | .LOG:  shutting down
postgres                     | LOG:  database system is shut down
postgres                     |  done
postgres                     | server stopped
postgres                     |
postgres                     | PostgreSQL init process complete; ready for start up.
postgres                     |
postgres                     | LOG:  database system was shut down at 2022-05-08 11:24:19 GMT
postgres                     | LOG:  MultiXact member wraparound protections are now enabled
postgres                     | LOG:  autovacuum launcher started
postgres                     | LOG:  database system is ready to accept connections
```

8. Reference Links :

- https://knative.dev/docs/
- https://serverlessworkflow.io/
- https://kogito.kie.org/
- https://github.com/kubernetes/dashboard/blob/master/docs/user/access-control/creating-sample-user.md
- https://www.youtube.com/watch?v=zx0_DIG6698
- https://vocon-it.com/2018/12/10/kubernetes-4-persistent-volumes-hello-world/
- https://dzone.com/refcardz/getting-started-with-quarkus-serverless-functions
- https://dzone.com/articles/bind-a-cloud-event-to-knative
- https://www.youtube.com/watch?v=zx0_DIG6698
- https://www.youtube.com/watch?v=69OfdJ5BIzs
- https://jamesdefabia.github.io/docs/user-guide/kubectl-overview/
- https://developers.redhat.com/blog/2020/06/30/kourier-a-lightweight-knative-serving-ingress#what_is_knative_
- https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard/
