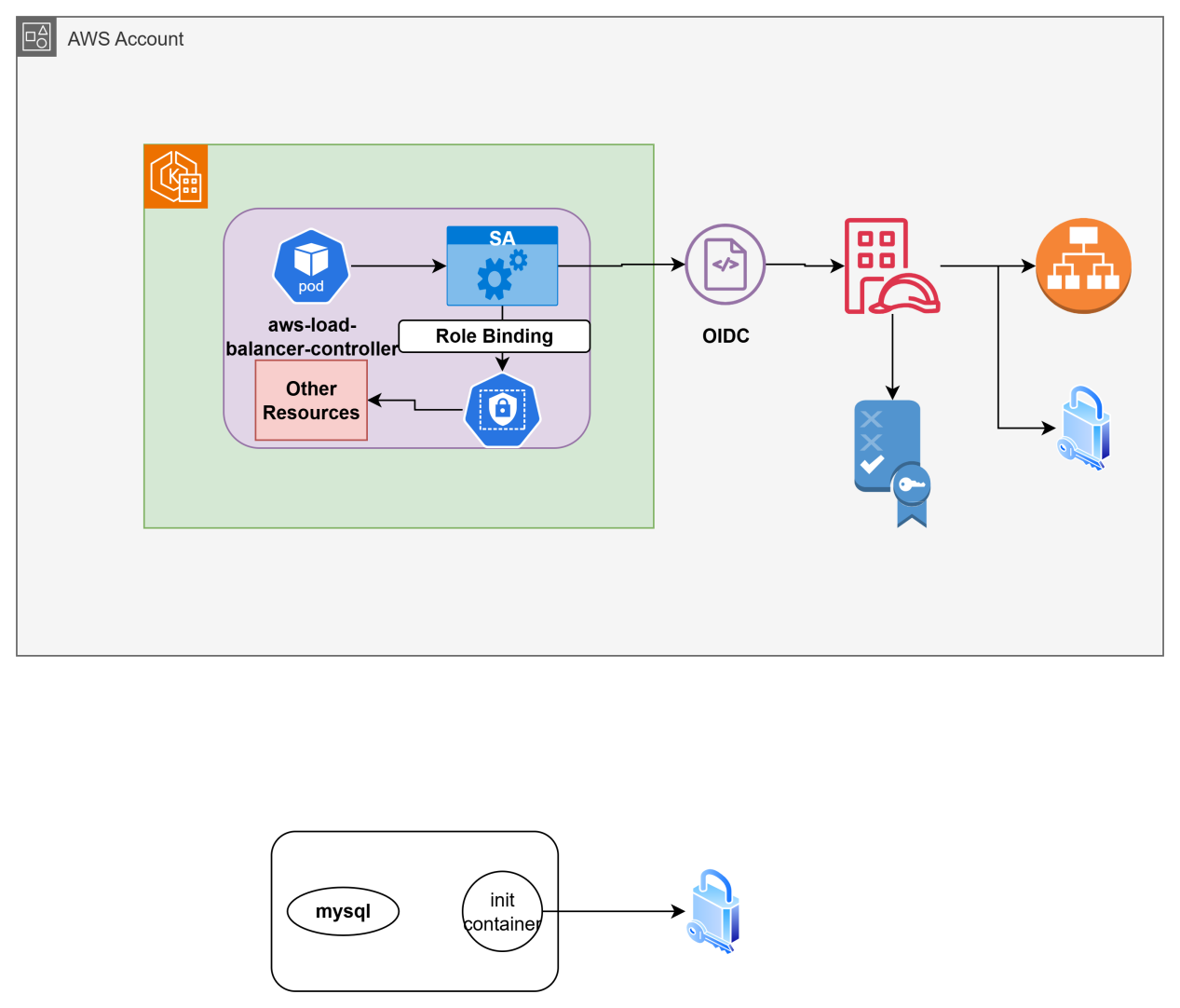
SESSION - 69

**RBAC**

--> Role based access control

--> AWS configuration map need to change for integration.

--> for this IAM role need to attach one file



**ServiceAccount**

**================**

non human rbac resource attached to pod access kube-api server and external resources provisioning like load balancer, accessing secrets, etc..

when we create a namespace kubernetes will create default sa. by default all pods run with default sa...

default sa will not have any permissions..

1. create OIDC provider

REGION\_CODE=us-east-1

CLUSTER\_NAME=roboshop-dev

ACC\_ID=315069654700

**--> kubens roboshop**

**--> kubectl get sa roboshop-mysql-secret-reader -o yaml**

**sa-create.yaml**

apiVersion: v1

kind: ServiceAccount

metadata:

annotations:

eks.amazonaws.com/role-arn: arn:aws:iam::315069654700:role/eksctl-roboshop-dev-addon-iamserviceaccount-r-Role1-z0lIcOetEsmx

labels:

app.kubernetes.io/managed-by: eksctl

name: roboshop-mysql-secret-reader

namespace: roboshop

**--> cd k8-rbac/**

**--> ls**

**--> kubectl apply -f sa.yaml**

**--> kubectl get pods**

**--> kubectl exec -it aws-cli -- bash**

**--> aws secretsmanager get-secret-value --secret-id roboshop/mysql/password**

**--> aws secretsmanager get-secret-value --secret-id roboshop/mysql/password --query secretstring --output text**

**service account**

**===============**

It is non human user that pod uses to run. by default when we create namespace a serviceaccount with name is default is created

1. Create OIDC provider

REGION\_CODE=us-east-1

CLUSTER\_NAME=roboshop-dev

ACC\_ID=315069654700

eksctl utils associate-iam-oidc-provider \

--region $REGION\_CODE \

--cluster $CLUSTER\_NAME \

--approve

2. Create policy and attach permissions

arn:aws:iam::315069654700:policy/RoboShopMySQLSecretReader

eksctl create iamserviceaccount \

--cluster=$CLUSTER\_NAME \

--namespace=roboshop \

--name=roboshop-mysql-secret-reader \

--attach-policy-arn=arn:aws:iam::315069654700:policy/RoboShopMySQLSecretReader \

--override-existing-serviceaccounts \

--region $REGION\_CODE \

--approve

this command creates IAM role and SA and integrates them

pod

1. with in the cluster

2. outside of the cluster... secretmanager IAM Role and Policy

SA --> Role and Policy attach

aws secretsmanager get-secret-value --secret-id roboshop/mysql/password

1. get the secret values from secretsmanager

2. create a env variable with this value and key as MYSQL\_ROOT\_PASSWORD

**InitContainers**

**===============**

init containers before main containers run in kubernetes. they make sure all the dependencies are setup like fetching secrets, make sure other services are running, etc..

Init containers always run to completion

Each init container must complete successfully before the next one starts.

How you make the secret avaialable to main container...

--> Container inside pod can, you can share same network space and rage

init container fetch secret and store it inside file. main container can get read the file and set it as env variable...

--> if I’m not given anything to pod where it will store.

pods by default uses node storage. if pod gets deleted, this storage also will be deleted. so storage lies until pod lives. these are ephemeral volumes...

**K8-roboshop/mysql/secret-manifest.yaml**

apiVersion: apps/v1

kind: Deployment

metadata:

name: mysql

namespace: roboshop

# deployment labels

labels:

component: mysql

project: roboshop

tier: database

spec:

replicas: 1

# These are the labels replica set use to create pod replicas, this should match pod labels

selector:

matchLabels:

component: mysql

project: roboshop

tier: database

# This is pod definition

template:

metadata:

labels:

component: mysql

project: roboshop

tier: database

spec:

serviceAccount: roboshop-mysql-secret-reader

volumes:

- name: mysql-secret

emptyDir: {}

containers:

- name: mysql

image: shan2324/mysql:v2

imagePullPolicy: Always

volumeMounts:

- mountPath: /tmp

name: mysql-secret

initContainers:

- name: fetch-secret

image: amazon/aws-cli

command:

- sh

- -c

- |

aws secretsmanager get-secret-value --secret-id roboshop/mysql/password --query SecretString --output text | jq -r .MYSQL\_ROOT\_PASSWORD > /tmp/mysql\_root\_password.txt

volumeMounts:

- mountPath: /tmp

name: mysql-secret

---

apiVersion: v1

kind: Service

metadata:

name: mysql

namespace: roboshop

labels:

component: mysql

project: roboshop

tier: database

spec:

selector:

component: mysql

project: roboshop

tier: database

ports:

- protocol: TCP

port: 3306 # service port

targetPort: 3306 # container port

**--> aws secretsmanager get-secret-value --secret-id roboshop/mysql/password --query secretstring --output text | jq -r .MYSQL\_ROOT\_PASSWORD**

**--> aws secretsmanager get-secret-value --secret-id roboshop/mysql/password --query secretstring --output text | jq -r .MYSQL\_ROOT\_PASSWORD** > **/tmp/mysql\_root\_password.txt**

**--> cd /tmp/**

**--> cat mysql\_root\_password.txt**

**-->** push and pull the code

**--> cd k8-roboshop/mysql**

**--> kubectl apply -f secret-mainfest.yaml**

**--> kubectl get pods**

**--> kubectl logs log-name**(get it from pods)

**--> kubectl describe pod log-name**

**--> kubectl logs log-name**

**--> kubectl get pods**

**--> kubectl delete deployment mysql**

**ENTRY POINT:**

No body can override arguments you can only supply to the through CMD.

**roboshop-docker/mysql/entrypoint.sh**

#!/bin/bash

if [ -f /tmp/mysql\_root\_password.txt ]; then

PASSWORD=$(cat /tmp/mysql\_root\_password.txt)

echo "Accessed Root password"

else

echo "Password file not found"

exit 1

fi

# Making it as available in env

export MYSQL\_ROOT\_PASSWORD=$PASSWORD

rm -rf /tmp/mysql\_root\_password.txt

exec /entrypoint.sh mysqld

**Roboshop-docker/mysql/Dockerfile**

#FROM mysql:8.0

#ENV MYSQL\_ROOT\_PASSWORD=RoboShop@1

#COPY db/\* /docker-entrypoint-initdb.d/

FROM mysql:8.0

COPY db/\* /docker-entrypoint-initdb.d/

COPY entrypoint.sh /usr/local/bin/entrypoint.sh

RUN chmod +x /usr/local/bin/entrypoint.sh

ENTRYPOINT ["/usr/local/bin/entrypoint.sh"]

--> push and pull the code

**cd roboshop-docker/mysql**

**docker build -t shan2324/mysql:v2 .**

**--> docker push shan2324/mysql:v2**

**Image pull policy**

The imagePullPolicy for a container and the tag of the image both affect when the [kubelet](https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/) attempts to pull (download) the specified image.

Here's a list of the values you can set for imagePullPolicy and the effects these values have:

**IfNotPresent**

the image is pulled only if it is not already present locally.

--> **k8-roboshop/mysql**

--> **kubectl apply -f manifest.yaml**

**--> kubectl get pods**

**--> kubectl exec -it log-name -- bash**

**--> mysql -u root -pRoboShop@1**

**--> show databases;**

**Exit**

**--> kubectl delete log-name**

**--> kubectl delete pod log-name**

**--> kubectl delete -f manifest.yaml**

**--> git pull**

**--> kubectl apply -f secret-manifest.yaml**

**-->**

**EMPTY DIRECTROY**

--> If you create empty directory volume all the containers inside the pods it can access it. It is an empheral volume.

then we need to use volumes...

1. empheral volumes --> temp storage required by pod

\* emptyDir --> if you want to share storage to all the containers inside pod, we can go for emptyDir

\* hostPath

2. external volumes --> by default we made is retain

mysql image

Projects achievements

--> earlier we were using kubernates secrets, I changed from kubernates secrets to AWS secrete manager.

--> How you did - by using unit container

What is unit containers?

Ans))

FROM mysql:8.0

COPY \*.sql /init.d/

eksctl utils associate-iam-oidc-provider \

--region $REGION\_CODE \

--cluster $CLUSTER\_NAME \

--approve

2. create IAM Policy to access secret manager

3. create K8 service account, IAM Role and attach policy to the role using eksctl command

eksctl create iamserviceaccount \

--cluster=$CLUSTER\_NAME \

--namespace=roboshop \

--name=roboshop-mysql-secret-reader \

--attach-policy-arn=arn:aws:iam::315069654700:policy/RoboShopMySQLSecretReader \

--override-existing-serviceaccounts \

--region $REGION\_CODE \

--approve