SESSION - 68

--> ingres controller

--> in running kubernates applications it will use for external access provide.

--> ingress controller is the load balancer that is traffic through the cluster ingress is the resource that creating the routing rules.it is send traffic to the service and service to the pods.

**RBAC**

**=====**

Role based access control

Authentication and Authorization

Nouns --> names

Verbs --> what action we do

AWS:

Services -> EC2, R53, VPC, etc

Verbs --> createInstance, GetInstance, UpdateInstance, DeleteInstance

Kubernates:

createPod, ReadPod, UpdatePod, DeletePod

Roles Permissions

Trainee read all resources in roboshop

Junior create Pod

Senior Create Update

Team Lead Delete access

Role and RoleBinding(Which role is binded to which user)

IAM, RBAC in K8

Suresh just joined team, he wants to access his namespace in EKS.

1. create user in IAM

2. He need to list his cluster. Create a policy to describe the cluster

3. we need to integrate IAM user with K8 Role

--> Create a k8-rnac directory

**role.yaml**

apiVersion: rbac.authorization.k8s.io/v1

kind: Role

metadata:

namespace: roboshop

name: roboshop-trainee

rules:

- apiGroups: [""] # "" indicates the core API group

resources: ["pods"] # nouns

verbs: ["get", "watch", "list"] # actions

---

apiVersion: rbac.authorization.k8s.io/v1

kind: Role

metadata:

namespace: roboshop

name: roboshop-junior

rules:

- apiGroups: [""] # "" indicates the core API group

resources: ["pods"] # nouns

verbs: ["get", "watch", "list","create"] # actions

- apiGroups: ["apps"]

#

# at the HTTP level, the name of the resource for accessing Deployment

# objects is "deployments"

resources: ["deployments"]

verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]

---

--> Go to IAM --> create user--> user name your wish --> next --> create user

--> present permissions not giving but we need to give a permission describe the cluster.

--> Go to policies --> create policy --> the policy access to EKS services --> in read describe cluster

--> we want to describe one cluster specifically.

--> take cluster ARN and give access this one alone.

--> region - us-east-1 --> name=roboshop-dev --> ADD ARN’s

--> policy details --> RoboshopEKSDescribe --> create policy aws sts get-caller-identity

aws eks update-kubeconfig --region us-east-1 --name roboshop-dev

--> video time 16:00 any doubt please gone through video.

--> Go to users and give created policy.

--> add permissions --> attach policies --> RoboshopEKSDescribe --> next --> add permissions.

--> roboshop cluster you can describe that enough for authentication.

--> roboshop team have saparate instance.

--> create instance user have access that instance

--> this roboshop instance need to login user

--> eks admin

--> Connect the server

--> aws configure --> add credentials

--> user just joined team, he wants to access his namespace in EKS.

**--> kubectl create namespace roboshop**

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

**--> kubectl get role -n roboshop**

**--> kubectl get rolebinding -n roboshop**

**--> kubectl get configmap aws-auth -n kube-system -o yaml**

**aws-auth.yaml**

apiVersion: v1

data:

mapRoles: |

- rolearn: arn:aws:iam::315069654700:role/eksctl-roboshop-dev-nodegroup-robo-NodeInstanceRole-TtRghdCKdBjV

groups:

- system:bootstrappers

- system:nodes

username: system:node:{{EC2PrivateDNSName}}

mapUsers: |

- groups:

- roboshop-trainee

userarn: arn:aws:iam::315069654700:user/suresh

username: suresh

kind: ConfigMap

metadata:

creationTimestamp: "2025-08-06T01:47:38Z"

name: aws-auth

namespace: kube-system

uid: 7c47ebd8-d158-4e4e-b0e3-dce2584f27a6

**--> git pull**

**--> kubectl apply -f aws-auth.yaml**

**--> kubectl get configmap aws-auth -n kube-system -o yaml**

**--> curl -O <https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.3/2025-08-03/bin/linux/amd64/kubectl>**

**--> chmod +x ./kubectl**

**--> sudo cp kubectl /usr/local/bin/kubectl**

**--> kubectl get pods**

**-->** kubectl not working need to authenticate.

**--> aws sts get-caller-identity** (login information)

**--> aws eks update-kubeconfig --region us-east-1 --name roboshop-dev**

**--> ls -la**

**--> kubectl get pods**

**role.yaml**

apiVersion: rbac.authorization.k8s.io/v1

kind: Role

metadata:

namespace: roboshop

name: roboshop-trainee

rules:

- apiGroups: [""] # "" indicates the core API group

resources: ["pods"] # nouns

verbs: ["get", "watch", "list"] # actions

---

apiVersion: rbac.authorization.k8s.io/v1

kind: Role

metadata:

namespace: roboshop

name: roboshop-junior

rules:

- apiGroups: [""] # "" indicates the core API group

resources: ["pods"] # nouns

verbs: ["get", "watch", "list","create"] # actions

- apiGroups: ["apps"]

#

# at the HTTP level, the name of the resource for accessing Deployment

# objects is "deployments"

resources: ["deployments"]

verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]

---

apiVersion: rbac.authorization.k8s.io/v1

kind: Role

metadata:

namespace: roboshop

name: roboshop-lead

rules:

- apiGroups: ["\*"] # "" indicates the core API group

resources: ["\*"] # nouns

verbs: ["\*"] # actions

---

apiVersion: rbac.authorization.k8s.io/v1

# This role binding allows "jane" to read pods in the "default" namespace.

# You need to already have a Role named "pod-reader" in that namespace.

kind: RoleBinding

metadata:

name: suresh

namespace: roboshop

subjects:

# You can specify more than one "subject"

- kind: User

name: suresh # "name" is case sensitive

apiGroup: rbac.authorization.k8s.io

roleRef:

# "roleRef" specifies the binding to a Role / ClusterRole

kind: Role #this must be Role or ClusterRole

name: roboshop-lead # this must match the name of the Role or ClusterRole you wish to bind to

apiGroup: rbac.authorization.k8s.io

---

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRole

metadata:

name: roboshop-cluster

rules:

- apiGroups: [""]

resources: ["persistentvolumes"]

verbs: ["get", "list", "watch"]

---

apiVersion: rbac.authorization.k8s.io/v1

# This cluster role binding allows anyone in the "manager" group to read secrets in any namespace.

kind: ClusterRoleBinding

metadata:

name: roboshop-cluster

subjects:

- kind: User

name: suresh # Name is case sensitive

apiGroup: rbac.authorization.k8s.io

roleRef:

kind: ClusterRole

name: roboshop-cluster

apiGroup: rbac.authorization.k8s.io

**--> git clone k8-roboshop.git**

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

**--> ls**

**--> cd mongodb/**

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

**--> kubectl get pvc**

**--> kubectl get pv**

**--> kubectl get api-resources | grep pers**

**--> kubectl api-resources | grep pers**

**--> kubectl get pvc -n roboshop**

--> PV is a cluster level.

**--> kubectl get pv**

--> like this we will give rule,rule binding,cluster,cluster binding.

**service account**

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

**--> kubectl get sa -n roboshop**

**-->** if one name space created one name service also will create.

--> kubectl describe sa default -n roboshop

**--> kubectl get pods -n roboshop**

**--> kubectl describe pod filename -n roboshop**

--> we need to create one service account and provide access it.

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

--> AWS secrets manager --> secrets --> store a new secret --> type=other type --> key pairs MYSQL\_ROOT\_PASSWORD --> Roboshop@1

--> secreate name = roboshop/mysql/password --> next --> store

--> One secrete was created.

--> create one policy

--> IAM --> Policies --> Create policy --> secret = secrets manager --> Read = Get secret value

--> specifically you can access this URL.

--> Policy name = RoboshopMYSQLsecretReader--> create policy

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 -- this is policy ARN

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

**--> kubectl get sa -n roboshop**

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

this command creates IAM role and SA and integrates them

--> This automatically created AWS

--> This role will create and map to service account

**sa.yaml**

apiVersion: rbac.authorization.k8s.io/v1

kind: Role

metadata:

namespace: roboshop

name: sa-reader

rules:

- apiGroups: ["\*"] # "" indicates the core API group

resources: ["\*"] # nouns

verbs: ["get", "watch", "list"] # actions

---

apiVersion: rbac.authorization.k8s.io/v1

# This role binding allows "jane" to read pods in the "default" namespace.

# You need to already have a Role named "pod-reader" in that namespace.

kind: RoleBinding

metadata:

name: sa-reader

namespace: roboshop

subjects:

# You can specify more than one "subject"

- kind: ServiceAccount

name: roboshop-mysql-secret-reader # "name" is case sensitive

namespace: roboshop

roleRef:

# "roleRef" specifies the binding to a Role / ClusterRole

kind: Role #this must be Role or ClusterRole

name: sa-reader # this must match the name of the Role or ClusterRole you wish to bind to

apiGroup: rbac.authorization.k8s.io

---

apiVersion: v1

kind: Pod

metadata:

name: aws-cli

namespace: roboshop

labels:

purpose: sa

spec:

serviceAccount: roboshop-mysql-secret-reader

containers:

- name: awscli

image: amazon/aws-cli

command: ["sleep","10000"]

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

**--> kubectl get pods**

**--> kubectl get pods -n roboshop**

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