**CAPESTONE PROJECT-1**

**POST GRADUATE PROGRAM IN DEVOPS**

**NAME : SIDDOJU RAVI MOHAN**

**MAIL : ravimohans1@gmail.com**

**Mobile :7337056757**

**https://github.com/siddojuravi/ravimohan-capstone-project.git**

**Infra Optimization.**

Course-end Project 1

DESCRIPTION

Create a DevOps infrastructure for an e-commerce application to run on high-availability mode.

**Background of the problem statement:**  
A popular payment application, **EasyPay**where users add money to their wallet accounts, faces an issue in its payment success rate. The timeout that occurs with  
the connectivity of the database has been the reason for the issue.  
While troubleshooting, it is found that the database server has several downtime instances at irregular intervals. This situation compels the company to create their own infrastructure that runs in high-availability mode.  
Given that online shopping experiences continue to evolve as per customer expectations, the developers are driven to make their app more reliable, fast, and secure for improving the performance of the current system.

**Implementation requirements:**

1. Create the cluster (EC2 instances with load balancer and elastic IP in case of AWS)
2. Automate the provisioning of an EC2 instance using Ansible or Chef Puppet
3. Install Docker and Kubernetes on the cluster
4. Implement the network policies at the database pod to allow ingress traffic from the front-end application pod
5. Create a new user with permissions to create, list, get, update, and delete pods
6. Configure application on the pod
7. Take snapshot of ETCD database
8. Set criteria such that if the memory of CPU goes beyond 50%, environments automatically get scaled up and configured

**The following tools must be used:**

1. EC2
2. Kubernetes
3. Docker
4. Ansible or Chef or Puppet

**The following things to be kept in check:**

1. You need to document the steps and write the algorithms in them.
2. The submission of your GitHub repository link is mandatory. In order to track your tasks, you need to share the link of the repository.
3. Document the step-by-step process starting from creating test cases, then executing them, and recording the results.
4. You need to submit the final specification document, which includes:

* Project and tester details
* Concepts used in the project
* Links to the GitHub repository to verify the project completion
* Your conclusion on enhancing the application and defining the USPs (Unique Selling Points)

**5. To create a new user with permissions to create, list, get, update, and delete pods in Kubernetes**

**Create a new user account with a unique username and password.**

#kubectl create serviceaccount myuser

#kubectl create role myrole --verb=create,list,get,update,delete --resource=pods

#kubectl create rolebinding myuser-binding --role=myrole --serviceaccount=default:myuser

#kubectl run mypod --image=nginx --serviceaccount=myuser

**$vim pod-admin.yaml**

**Create a new role with the required permissions. For example, you can create a role named "pod-admin" with the following permissions:**

apiVersion: rbac.authorization.k8s.io/v1

kind: Role

metadata:

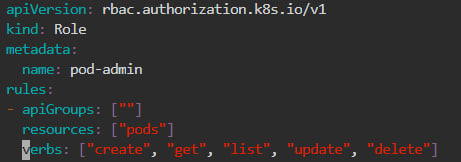
name: pod-admin

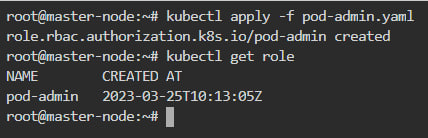
rules:

- apiGroups: [""]

resources: ["pods"]

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





**Bind the role to the user account using a RoleBinding. For example, you can create a RoleBinding named "pod-admin-binding" to associate the "pod-admin" role with the user "newuser".**

**#vim pod-admin-binding**

apiVersion: rbac.authorization.k8s.io/v1

kind: RoleBinding

metadata:

name: pod-admin-binding

subjects:

- kind: User

name: newuser

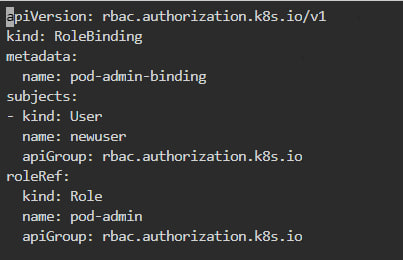
apiGroup: rbac.authorization.k8s.io

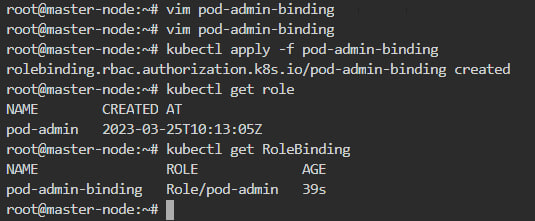
roleRef:

kind: Role

name: pod-admin

apiGroup: rbac.authorization.k8s.io





**Once you have created the user, role, and role binding, the user will have the necessary permissions to create, list, get, update, and delete pods in the Kubernetes cluster.**