**Deploy application using Kubernetes Dashboard**

**Step 1: Create namespace and service account.**

$k create ns cep-project1

$k create sa sandry -n cep-project1

|  |
| --- |
| **Vi Account.yaml**  apiVersion: v1  kind: ServiceAccount  metadata:  name: sandry  namespace: cep-project1 |

**$Kubectl apply -f account.yaml**

**$kubectl get sa -n cep-project1**

**Step 2:**

**Assign cluster admin role by creating clusterRole and ClusterRoleBinding**

|  |
| --- |
| **Vi sandry-clusterRole.yaml**  apiVersion: rbac.authorization.k8s.io/v1  kind: ClusterRoleBinding  metadata:  namespace: cep-project1  name: sandry  roleRef:  apiGroup: rbac.authorization.k8s.io  kind: ClusterRole  name: cluster-admin  subjects:  - kind: ServiceAccount  name: sandry  namespace: cep-project1 |

Now Sandry is a cluster admin so he has access to run Kubernetes dashboard.

**We need sandry account secret key to access Kubernetes dashboard**

$kubectl describe secret -n kube-system

**Step 2: Deploy Kubernetes dashboard:**

Deploy Kubernetes dashboard by running recommended YAML file. And launch dashboard with service account secret key token.

$kubectl apply -f [**https://raw.githubusercontent.com/kubernetes/dashboard/v2.5.0/aio/deploy/recommended.yaml**](https://raw.githubusercontent.com/kubernetes/dashboard/v2.5.0/aio/deploy/recommended.yaml)

Edit service from clusterIP to Nodeport to use access kubenetes dashboard outside cluster.

$kubectl edit svc Kubernetes-dashboard -n Kubernetes-dashboard.

Now we cam access it by node IP with static port.

**Step 3: Configuration YAML files for PHP/MYSQL application.**

1. First launch mysql deployment with mysql-deployment.

Create Persistent volume and persistence volume claim to mount pod data. Ref. Mysql-storage.yaml

Create mysql-secret.yaml to create encoded password. Ref mysql-secret.yaml

Create cluster service to connect to mysql Database within cluster. Ref mysql-svc.yaml

Finally create mysql deployment. Ref Mysql-deployment.yaml

Execute mysql pod and create table users and add table data.

1. Launch PHP-Web application to connect to mysql database.

Create persistent volume and persistent volume claim. Ref web-storage.yaml

Create configmap to mount index.php file at /var/www/html. Ref Exfile

Create NodePort service to connect to Web from outside. Ref web-service.yaml

Create web deployment. Ref web-deployment.yaml

1. Now we can connect to web application with Node IP on NodePort number.

$curl IP:NodePort/index.php

|  |
| --- |
| 1. **mysql-secret.yaml**   apiVersion: v1  kind: Secret  metadata:  name: mysql-secret  type: kubernetes.io/basic-auth  stringData:  password: test1234   1. **mysql-storage.yaml**   apiVersion: v1  kind: PersistentVolume  metadata:  name: mysql-pv-volume  labels:  type: local  spec:  storageClassName: manual  capacity:  storage: 20Gi  accessModes:  - ReadWriteOnce  hostPath:  path: "/mnt/data"  ---  apiVersion: v1  kind: PersistentVolumeClaim  metadata:  name: mysql-pv-claim  spec:  storageClassName: manual  accessModes:  - ReadWriteOnce  resources:  requests:  storage: 20Gi   1. **mysql-deployment.yaml**   apiVersion: apps/v1  kind: Deployment  metadata:  name: mysql  namespace: cep-project1  spec:  selector:  matchLabels:  app: mysql  strategy:  type: Recreate  template:  metadata:  labels:  app: mysql  spec:  containers:  - image: mysql:5.6  name: mysql  env:  - name: MYSQL\_ROOT\_PASSWORD  valueFrom:  secretKeyRef:  name: mysql-secret  key: password  ports:  - containerPort: 3306  name: mysql  volumeMounts:  - name: mysql-persistent-storage  mountPath: /var/lib/mysql  volumes:  - name: mysql-persistent-storage  persistentVolumeClaim:  claimName: mysql-pv-claim   1. **Mysql-svc.yaml**   apiVersion: v1  kind: Service  metadata:  name: mysql  spec:  ports:  - port: 3306  selector:  app: mysql |

|  |
| --- |
| 1. **web-storage.yaml**   apiVersion: v1  kind: PersistentVolume  metadata:  name: web-pv-volume  labels:  type: local  spec:  storageClassName: manual  capacity:  storage: 5Gi  accessModes:  - ReadWriteOnce  hostPath:  path: "/mnt/"  ---  apiVersion: v1  kind: PersistentVolumeClaim  metadata:  name: web-pv-claim  spec:  storageClassName: manual  accessModes:  - ReadWriteOnce  resources:  requests:  storage: 5Gi   1. **Create configMap**   $kubectl create cm exfile –file-name = index.php   1. **web-deployment.yaml**   apiVersion: apps/v1  kind: Deployment  metadata:  name: myphp-deployment  namespace: cep-project1  spec:  replicas: 2  selector:  matchLabels:  env: production-frontend  template:  metadata:  name: myfrontend-pod  labels:  env: production-frontend  spec:  volumes:  - name: front-vol  persistentVolumeClaim:  claimName: web-pv-claim  - name: webfile  configMap:  name: exfile  containers:  - name: frontend  image: ragh19/phpproject:web\_v1  volumeMounts:  - mountPath: /var/www/html  name: front-vol  - mountPath: /var/www/html/index.php  subPath: index.php  name: webfile   1. **web-svc.yaml #nodePort service so that I can access application externally.**   apiVersion: v1  kind: Service  metadata:  name: myphp-service  spec:  type: NodePort  ports:  - targetPort: 80  port: 80  nodePort: 30009  selector:  env: production-frontend |

|  |
| --- |
| 1. **Index.php**   <?php  $servername = "mysql";  $database = "mydb";  $username = "root";  $password = "test1234";  // Create connection  $conn = mysqli\_connect($servername, $username, $password, $database);  // Check connection  if ($conn->connect\_error) {  die("Connection failed: " . $conn->connect\_error);  }  echo “Connected successfully”;  mysqli\_close($conn);  ?> |

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated