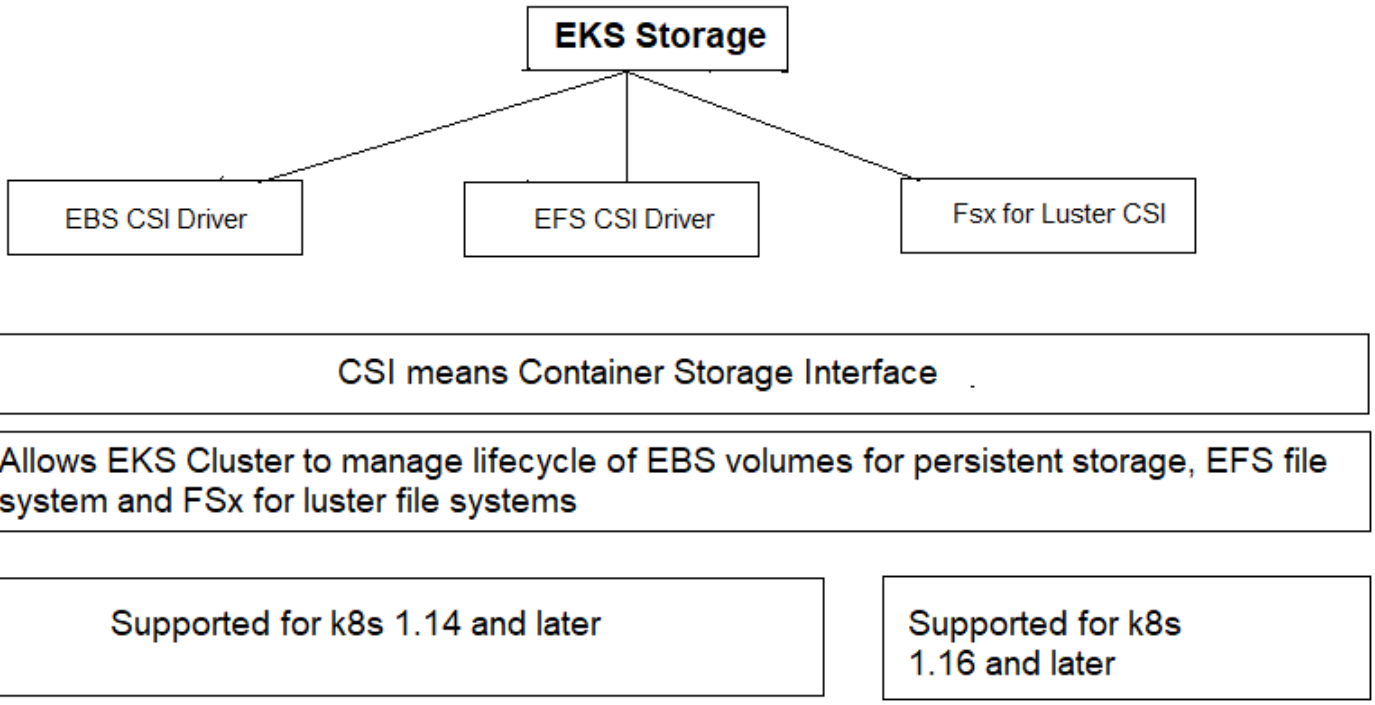
EKS Storage Class

EKS Storage Class Introduction

A StorageClass is a Kubernetes object that stores information about creating a persistent volume for your pod. With a StorageClass, you do not need to create a persistent volume separately before claiming it.

A StorageClass is a Kubernetes resource that enables dynamic storage provisioning. The administrator configures the StorageClass, which can then no longer be modified. First the StorageClass is created, then the PersistentVolumeClaim and finally the Pod. When a PVC is created, Kubernetes creates a PersistentVolume and binds it to the PVC automatically, depending on the VolumeBindingMode used in the StorageClass configuration. These three Kubernetes objects are required to check the test case of a StorageClass.

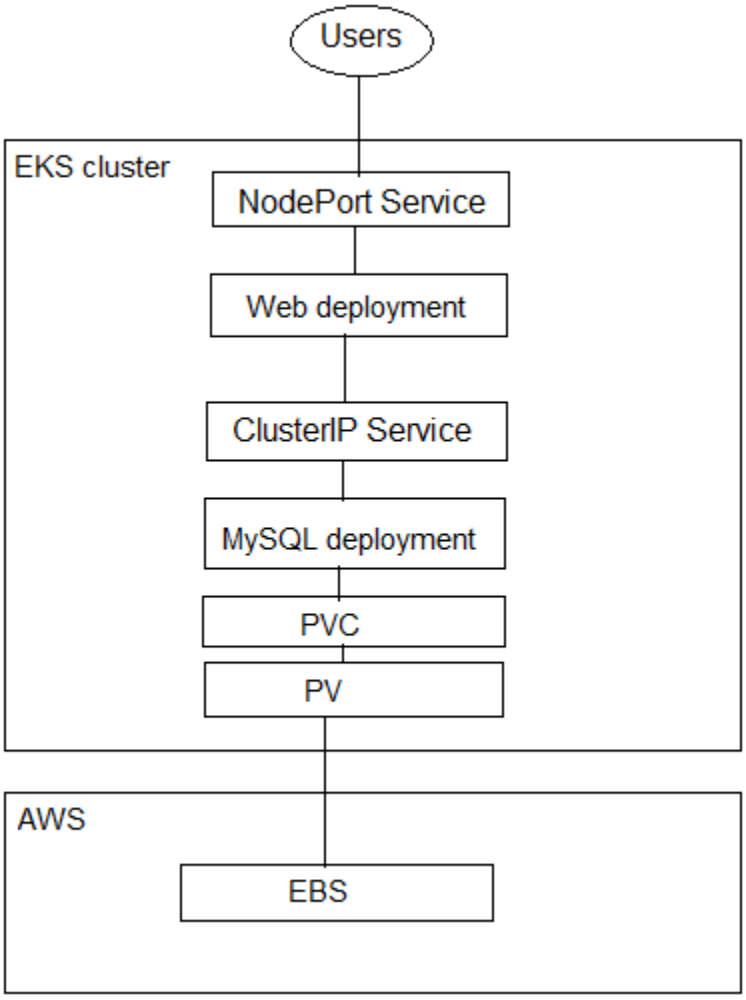


EKS hosted application storage with AWS EBS

Introduction

- EBS provides block level storage volumes for use with EC2 and Container instances
- We can mount these volumes as devices on our EC2 and Container instances.
- EBS volumes that are attached to an instance are exposed as storage volumes that persist independently from the life of the EC2 or Container instance.
- We can dynamically change the configuration of a volume attached to an instance.
- AWS recommends EBS for data that must be quickly accessible and requires long-term persistence.
- EBS is well suited to both database-style applications that rely on random reads and writes, and to throughput-intensive applications that perform long, continuous reads and writes.

Diagram



Topics

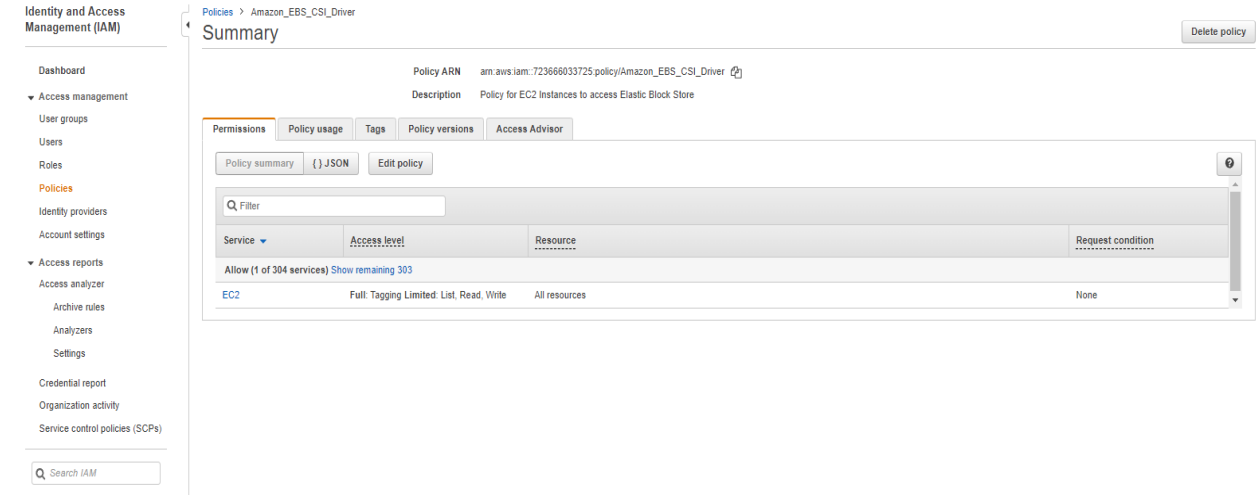
1. Install EBS CSI Driver.
2. Create Storage class, PVC, ConfigMap, MySQL Database Deployment & ClusterIP Service.
3. Create User Management Microservice Deployment & NodePort Service.

1. Install EBS CSI Driver.

- Go to Services -> IAM
- Create a Policy
  - Select JSON tab and copy paste the below JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:AttachVolume",
        "ec2:CreateSnapshot",
        "ec2:CreateTags",
        "ec2:CreateVolume",
        "ec2:DeleteSnapshot",
        "ec2:DeleteTags",
        "ec2:DeleteVolume",
        "ec2:DescribeInstances",
        "ec2:DescribeSnapshots",
        "ec2:DescribeTags",
        "ec2:DescribeVolumes",
        "ec2:DetachVolume"
      ],
      "Resource": "*"
    }
  ]
}
```

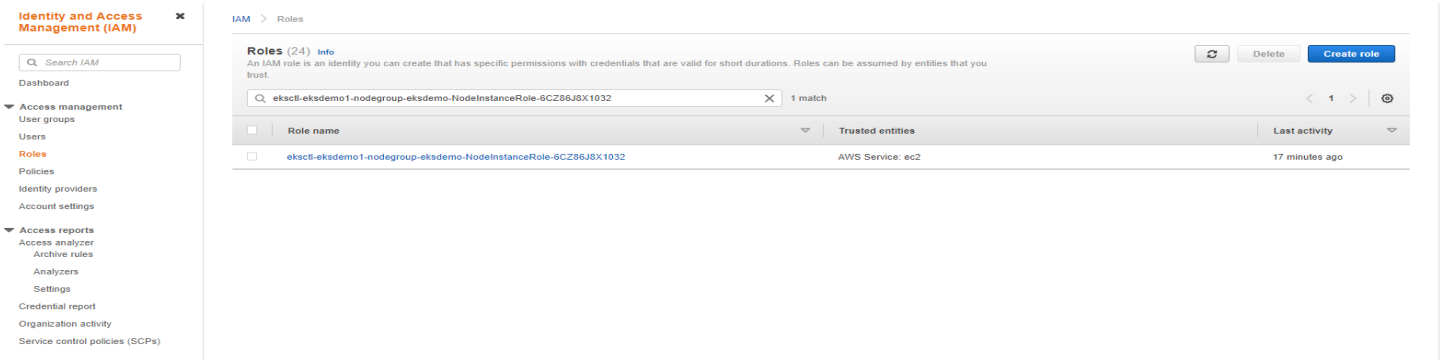
- Review the same in Visual Editor
- Click on Review Policy
- Name: Amazon_EBS_CSI_Driver
- Description: Policy for EC2 Instances to access Elastic Block Store
- Click on Create Policy

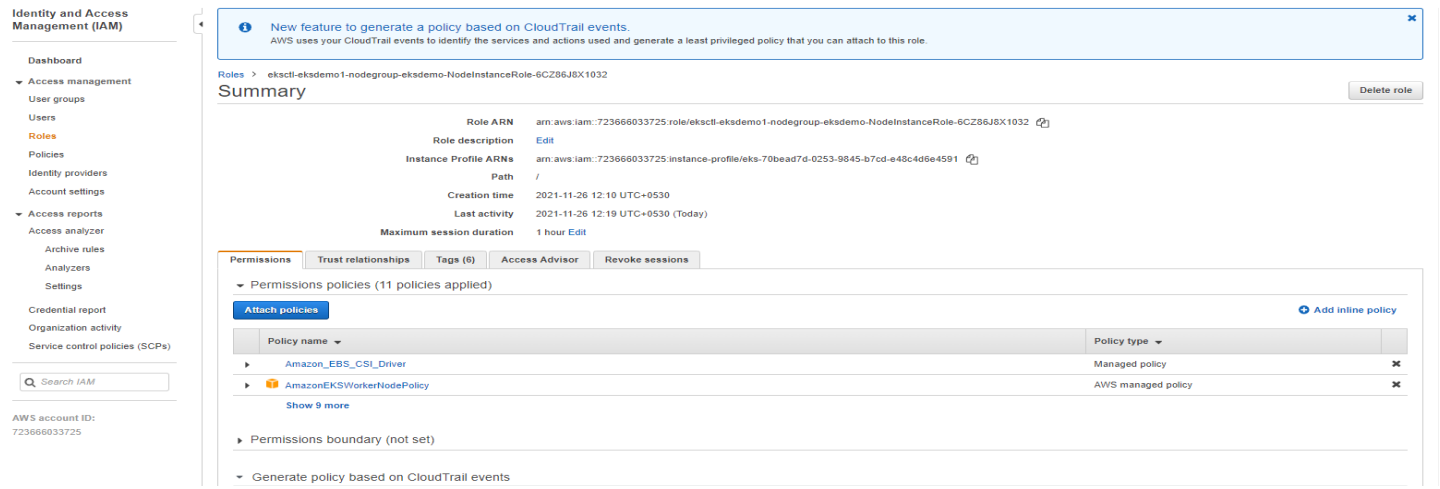


- Get the IAM role Worker Nodes using and associate this policy to that role

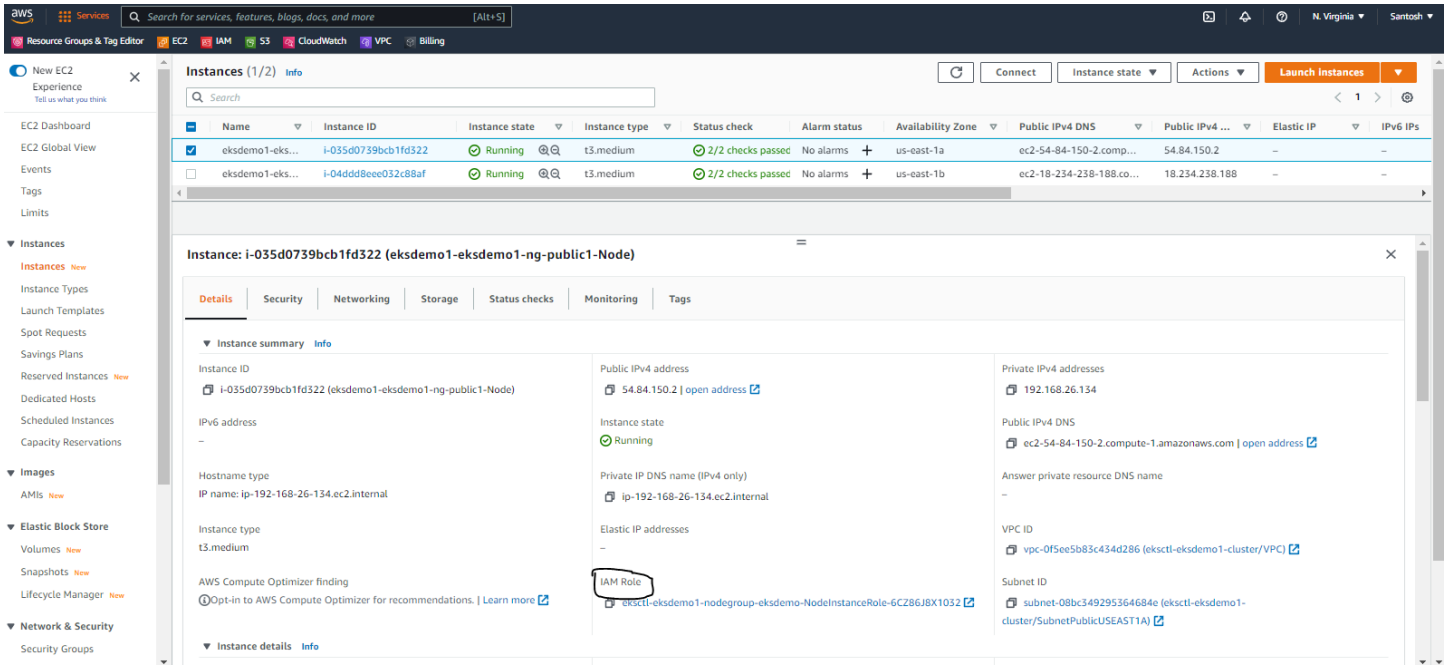
```
[root@localhost ~]# kubectl -n kube-system describe configmap aws-auth
Name:         aws-auth
Namespace:    kube-system
Labels:       <none>
Annotations:  <none>

Data
====
mapRoles:
----
- groups:
  - system:bootstrappers
  - system:nodes
  rolearn: arn:aws:iam::723666033725:role/eksctl-eksdemo1-nodegroup-eksdemo-NodeInstanceRole-6CZ86J8X1032
  username: system:node:{{EC2PrivateDNSName}}

Events:  <none>
[root@localhost ~]#
```

- Go to Services -> IAM -> Roles
- Search for role with name eksctl-eksdemo1-nodegroup and open it
- Click on Permissions tab
- Click on Attach Policies
- Search for Amazon_EBS_CSI_Driver and click on Attach Policy





This IAM role is attached to EKS cluster instances.



- Deploy Amazon EBS CSI Driver

```
[root@localhost ~]# kubectl apply -k "github.com/kubernetes-sigs/aws-ebs-csi-driver/deploy/kubernetes/overlays/stable/?ref=master"
serviceaccount/ebs-csi-controller-sa created
serviceaccount/ebs-csi-node-sa created
clusterrole.rbac.authorization.k8s.io/ebs-csi-node-role created
clusterrole.rbac.authorization.k8s.io/ebs-external-attacher-role created
clusterrole.rbac.authorization.k8s.io/ebs-external-provisioner-role created
clusterrole.rbac.authorization.k8s.io/ebs-external-resizer-role created
clusterrole.rbac.authorization.k8s.io/ebs-external-snapshotter-role created
clusterrolebinding.rbac.authorization.k8s.io/ebs-csi-attacher-binding created
clusterrolebinding.rbac.authorization.k8s.io/ebs-csi-node-getter-binding created
clusterrolebinding.rbac.authorization.k8s.io/ebs-csi-provisioner-binding created
clusterrolebinding.rbac.authorization.k8s.io/ebs-csi-resizer-binding created
clusterrolebinding.rbac.authorization.k8s.io/ebs-csi-snapshotter-binding created
deployment.apps/ebs-csi-controller created
Warning: policy/v1beta1 PodDisruptionBudget is deprecated in v1.21+, unavailable in v1.25+; use policy/v1 PodDisruptionBudget
poddisruptionbudget.policy/ebs-csi-controller created
daemonset.apps/ebs-csi-node created
csidriver.storage.k8s.io/ebs.csi.aws.com created
[root@localhost ~]#
```

```
[root@localhost ~]# kubectl get pods -n kube-system
NAME                                  READY   STATUS    RESTARTS   AGE
aws-node-8rdkj                        1/1     Running   1          30m
aws-node-9bv25                        1/1     Running   0          30m
coredns-66cb55d4f4-84ptq              1/1     Running   0          45m
coredns-66cb55d4f4-8v2xw              1/1     Running   0          45m
ebs-csi-controller-56dcb94444-q7wcj   6/6     Running   0          43s
ebs-csi-controller-56dcb94444-q825r   6/6     Running   0          43s
ebs-csi-node-jfkgc                    3/3     Running   0          41s
ebs-csi-node-zmlq9                    3/3     Running   0          41s
kube-proxy-5dmrc                      1/1     Running   0          30m
kube-proxy-wr4tb                      1/1     Running   0          30m
[root@localhost ~]#
```

2. Create Storage class, PVC, ConfigMap, MySQL Database Deployment & ClusterIP Service.
   - First create Storage class, PVC, ConfigMap

```
[root@localhost EBS]# cat storage_class.yml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ebs-sc
provisioner: ebs.csi.aws.com
volumeBindingMode: WaitForFirstConsumer

[root@localhost EBS]# cat pvc.yml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: ebs-mysql-pv-claim
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: ebs-sc
  resources:
    requests:
      storage: 4Gi

[root@localhost EBS]# cat configmap.yml
apiVersion: v1
kind: ConfigMap
metadata:
  name: usermanagement-dbcreation-script
data:
  mysql_usermgmt.sql: |-
    DROP DATABASE IF EXISTS usermgmt;
    CREATE DATABASE usermgmt;

[root@localhost EBS]#
```

```
[root@localhost EBS]# kubectl apply -f ../EBS/
configmap/usermanagement-dbcreation-script created
persistentvolumeclaim/ebs-mysql-pv-claim created
storageclass.storage.k8s.io/ebs-sc created
[root@localhost EBS]# kubectl get configmap
NAME                              DATA   AGE
kube-root-ca.crt                  1      60m
usermanagement-dbcreation-script  1      44s
[root@localhost EBS]# kubectl get pv
No resources found
[root@localhost EBS]# kubectl get pvc
NAME                STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   AGE
ebs-mysql-pv-claim  Pending                                      ebs-sc         55s
[root@localhost EBS]# kubectl get sc
NAME            PROVISIONER             RECLAIMPOLICY   VOLUMEBINDINGMODE      ALLOWVOLUMEEXPANSION   AGE
ebs-sc          ebs.csi.aws.com         Delete          WaitForFirstConsumer   false                  62s
gp2 (default)   kubernetes.io/aws-ebs   Delete          WaitForFirstConsumer   false                  61m
[root@localhost EBS]#
```

- Now create MySQL deployment and MySQL service.

```
[root@localhost EBS]# ls
configmap.yml  mysql_deployment.yml  mysql_service.yml  pvc.yml  storage_class.yml
[root@localhost EBS]# cat mysql_deployment.yml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mysql
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
        - name: mysql
          image: mysql:5.6
          env:
            - name: MYSQL_ROOT_PASSWORD
              value: dbpassword11
          ports:
            - containerPort: 3306
              name: mysql
          volumeMounts:
            - name: mysql-persistent-storage
              mountPath: /var/lib/mysql
            - name: usermanagement-dbcreation-script
              mountPath: /docker-entrypoint-initdb.d
      volumes:
        - name: mysql-persistent-storage
          persistentVolumeClaim:
            claimName: ebs-mysql-pv-claim
        - name: usermanagement-dbcreation-script
          configMap:
            name: usermanagement-dbcreation-script

[root@localhost EBS]#
```

```
[root@localhost EBS]# ls
configmap.yml  mysql_deployment.yml  mysql_service.yml  pvc.yml  storage_class.yml
[root@localhost EBS]# cat mysql_service.yml
apiVersion: v1
kind: Service
metadata:
  name: mysql
spec:
  selector:
    app: mysql
  ports:
    - port: 3306
  clusterIP: None

[root@localhost EBS]#
```

```
[root@localhost EBS]# ls
configmap.yml  mysql_deployment.yml  mysql_service.yml  pvc.yml  storage_class.yml
[root@localhost EBS]# kubectl apply -f ../EBS/
configmap/usermanagement-dbcreation-script unchanged
deployment.apps/mysql created
service/mysql created
persistentvolumeclaim/ebs-mysql-pv-claim unchanged
storageclass.storage.k8s.io/ebs-sc unchanged
[root@localhost EBS]# kubectl get sc
NAME            PROVISIONER             RECLAIMPOLICY   VOLUMEBINDINGMODE      ALLOWVOLUMEEXPANSION   AGE
ebs-sc          ebs.csi.aws.com         Delete          WaitForFirstConsumer   false                  5m23s
gp2 (default)   kubernetes.io/aws-ebs   Delete          WaitForFirstConsumer   false                  65m
[root@localhost EBS]# kubectl get pvc
NAME                STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
ebs-mysql-pv-claim   Bound    pvc-7112618b-22f6-4afe-afcf-bb3c11b3b806   4Gi        RWO            ebs-sc         5m29s
[root@localhost EBS]# kubectl get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                       STORAGECLASS   REASON   AGE
pvc-7112618b-22f6-4afe-afcf-bb3c11b3b806   4Gi        RWO            Delete           Bound    default/ebs-mysql-pv-claim   ebs-sc                  46s
[root@localhost EBS]# kubectl get pods

NAME                     READY   STATUS    RESTARTS   AGE
mysql-6fdd448876-xlsd6   1/1     Running   0          63s
[root@localhost EBS]#
[root@localhost EBS]#
```

- Deploy Amazon EBS CSI Driver and check the database is created or not.

```
[root@localhost EBS]# kubectl run -it --rm --image=mysql:5.6 --restart=Never mysql-client -- mysql -h mysql -pdbpassword11
If you don't see a command prompt, try pressing enter.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| #mysql50#lost+found |
| mysql              |
| performance_schema |
| usermgmt           |
+--------------------+
5 rows in set (0.00 sec)

mysql>
```

Now create a manifest of web deployment and its NodePort Service and deploy it.

```
[root@localhost EBS]# ls
configmap.yml  mysql_deployment.yml  mysql_service.yml  pvc.yml  storage_class.yml  web_deployment.yml  web_service.yml
[root@localhost EBS]# cat web_deployment.yml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: usermgmt-microservice
  labels:
    app: usermgmt-restapp
spec:
  replicas: 1
  selector:
    matchLabels:
      app: usermgmt-restapp
  template:
    metadata:
      labels:
        app: usermgmt-restapp
    spec:
      containers:
        - name: usermgmt-restapp
          image: stacksimplify/kube-usermanagement-microservice:1.0.0
          ports:
            - containerPort: 8095
          env:
            - name: DB_HOSTNAME
              value: "mysql"
            - name: DB_PORT
              value: "3306"
            - name: DB_NAME
              value: "usermgmt"
            - name: DB_USERNAME
              value: "root"
            - name: DB_PASSWORD
              value: "dbpassword11"

[root@localhost EBS]#
```

```
[root@localhost EBS]# ls
configmap.yml  mysql_deployment.yml  mysql_service.yml  pvc.yml  storage_class.yml  web_deployment.yml  web_service.yml
[root@localhost EBS]# cat web_service.yml
apiVersion: v1
kind: Service
metadata:
  name: usermgmt-restapp-service
  labels:
    app: usermgmt-restapp
spec:
  type: NodePort
  selector:
    app: usermgmt-restapp
  ports:
    - port: 8095
      targetPort: 8095
      nodePort: 31231

[root@localhost EBS]#
```

```
[root@localhost EBS]# kubectl apply -f ../EBS/
configmap/usermanagement-dbcreation-script unchanged
deployment.apps/mysql unchanged
service/mysql unchanged
persistentvolumeclaim/ebs-mysql-pv-claim unchanged
storageclass.storage.k8s.io/ebs-sc unchanged
deployment.apps/usermgmt-microservice created
service/usermgmt-restapp-service created
[root@localhost EBS]# kubectl  get pods
NAME                                    READY   STATUS    RESTARTS   AGE
mysql-6fdd448876-xlsd6                  1/1     Running   0          13m
usermgmt-microservice-6c6cdb5758-lj5dx  1/1     Running   0          21s
[root@localhost EBS]# kubectl  get svc
NAME                     TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
kubernetes               ClusterIP   10.100.0.1       <none>        443/TCP          79m
mysql                    ClusterIP   None             <none>        3306/TCP         13m
usermgmt-restapp-service NodePort    10.100.208.183   <none>        8095:31231/TCP   28s
[root@localhost EBS]#
```

Successfully deployed EKS hosted web application with EBS storage class!!