**Implementing Persistent Storage with EKS using EFS**

**1-Set up an EFS filesystem.**

**2. Integrate EFS with a Kubernetes deployment.**

**3. Test data persistence across pod restarts.**

**Prerequisites:**

**Install kubectl**

**Install eks**

**Install awscli**

**Add access key**

**Add roles for cluster and node permissions**

**Step1: Set up an EFS filesystem.**

**Open the Amazon EFS console.**

**Click Create file system.**

**Enter a name for your file system and select your desired performance mode.**

**Click Next.**

**Select the VPC and subnet where you want to create your file system.**

**Click Next.**

**Configure the security group for your file system.**

**Click Create.**

Once your file system is created, you will need to create an EFS access point. An EFS access point is a network interface that allows Kubernetes pods to access your EFS file system.

**STEP:2**

**Open the Amazon EFS console.**

**Click your file system.**

**Click Access points.**

**Click Create access point.**

**Enter a name for your access point.**

**Select the VPC and subnet where you want to create your access point.**

**Click Create.**

**Integrate EFS with a Kubernetes deployment.**

**To integrate EFS with a Kubernetes deployment, you will need to install the Amazon EFS CSI driver. The CSI driver is a Kubernetes plugin that allows Kubernetes to provision and manage EFS volumes.**

**Install the Helm package manager.**

**Add the Amazon EFS CSI driver Helm repository.**

**Install the Amazon EFS CSI driver.**

for install helm :

**https://phoenixnap.com/kb/install-helm**

**helm repo add aws-efs-csi-driver https://kubernetes-sigs.github.io/aws-efs-csi-driver/**

**helm install aws-efs-csi-driver aws-efs-csi-driver/aws-efs-csi-driver**

**Once the CSI driver is installed, you can verify that it is running by running the following command:**

| kubectl get pod -l app=aws-efs-csi-driver |
| --- |

[**https://docs.aws.amazon.com/eks/latest/userguide/efs-csi.html**](https://docs.aws.amazon.com/eks/latest/userguide/efs-csi.html)

**Once the CSI driver is installed, you can create a Kubernetes storage class for EFS. A storage class is a Kubernetes object that defines the type of storage that Kubernetes should provision when a pod requests a persistent volume.**

| **apiVersion: storage.k8s.io/v1 kind: StorageClass metadata:  name: efs-sc provisioner: efs.csi.aws.com parameters:  fsid: <YOUR\_EFS\_FILE\_SYSTEM\_ID>  provisioningMode: efs** |
| --- |

**Once you have created a storage class for EFS, you can create a persistent volume claim (PVC). A PVC is a Kubernetes object that requests a persistent volume from Kubernetes.**

| **apiVersion: v1 kind: PersistentVolumeClaim metadata:  name: efs-pvc spec:  accessModes:  - ReadWriteOnce  resources:  requests:  storage: 1Gi  storageClassName: efs-sc** |
| --- |

**Kubernetes will provision a persistent volume for you based on the storage class that you specified in the PVC.**

**Once the persistent volume is provisioned, you can create a Kubernetes deployment that uses the persistent volume.**

| **apiVersion: apps/v1 kind: Deployment metadata:  name: efs-deployment spec:  replicas: 2  selector:  matchLabels:  app: efs-deployment  template:  metadata:  labels:  app: efs-deployment  spec:  containers:  - name: efs-container  image: nginx:latest  volumeMounts:  - name: efs-volume  mountPath: /efs  volumes:  - name: efs-volume  persistentVolumeClaim:  claimName: efs-pvc** |
| --- |

**This deployment will create two pods that are each mounted to the EFS file system at the /efs path.**

**3. Test data persistence across pod restarts**

**To test data persistence across pod restarts, you can write a file to the EFS file system from one of the pods and then restart the pod.**

**Open a shell to one of the pods.**

**Create a file in the /efs directory.**

**Write some data to the file.**

**Restart the pod.**

**Once the pod has restarted, you can open a shell to the pod again and verify that the file is still present and that the data is still intact.**

**This demonstrates that data persisted to the EFS file system is available to pods even after the pods are restarted**

**(NOTE):**

**https://saturncloud.io/blog/using-aws-efs-access-points-to-mount-a-kubernetes-persistent-volume-in-eks-a-guide/**

**https://saturncloud.io/blog/using-aws-efs-access-points-to-mount-a-kubernetes-persistent-volume-in-eks-a-guide/**

**CONCLUSION of this lab:** The successful implementation of Elastic File System (EFS) as persistent storage within an Amazon EKS environment marks a significant stride in ensuring data durability, resilience, and seamless integration for containerized applications.

Throughout this lab, we accomplished the following key objectives:

We established an Elastic File System (EFS) within the AWS environment, providing a scalable and highly available storage solution capable of accommodating the dynamic storage needs of Kubernetes application

Integrating EFS with our Kubernetes deployment showcased the simplicity and effectiveness of leveraging EFS as a persistent storage solution within EKS clusters. The seamless integration eased the provisioning and consumption of persistent volumes by Kubernetes pods.

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