ELB AWS

Step 1: Create IAM Role using eksctl

Download an IAM policy for the AWS Load Balancer Controller that allows it to make calls to AWS APIs.

- Since the AWS Load Balancer Controller needs permissions to manage AWS Load Balancers, you need to create an IAM policy and associate it with a Kubernetes service account.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.11.0/docs/install/iam_policy.json
```

Create an IAM policy using the policy downloaded in the previous step.

- Creates a new IAM policy named AWSLoadBalancerControllerIAMPolicy.
- This policy allows the Load Balancer Controller to make AWS API calls.

```
aws iam create-policy \
   --policy-name AWSLoadBalancerControllerIAMPolicy \
   --policy-document file://iam_policy.json
```

Set up an IAM service account in an EKS cluster, allowing the AWS Load Balancer Controller to manage AWS Load Balancers on behalf of the Kubernetes cluster.

- Creates a Kubernetes ServiceAccount named aws-load-balancer-controller.
- Associates it with an IAM Role (AmazonEKSLoadBalancerControllerRole).
- Attaches the AWSLoadBalancerControllerIAMPolicy.
- Allows Kubernetes to use AWS IAM for authentication.

```
eksctl create iamserviceaccount \
  --cluster=alb-demo-cluster \
  --namespace=kube-system \
  --name=aws-load-balancer-controller \
  --role-name AmazonEKSLoadBalancerControllerRole \
  --attach-policy-arn=arn:aws:iam::<aws-account-id>:policy/AWSLoadBalancerControllerIAMPolicy \
  --region us-east-2 \
  --approve
```

Step 2: Install AWS Load Balancer Controller

Add the eks-charts Helm chart repository.

```
helm repo add eks https://aws.github.io/eks-charts
```

Update your local repo to make sure that you have the most recent charts (If Not first time).

```
helm repo update eks
```

Install the AWS Load Balancer Controller.

- Installs the AWS Load Balancer Controller in the kube-system namespace.
- Links it to the existing aws-load-balancer-controller service account.

```
helm install aws-load-balancer-controller eks/aws-load-balancer-controller \
  -n kube-system \
  --set clusterName=alb-demo-cluster \
  --set serviceAccount.create=false \
  --set serviceAccount.name=aws-load-balancer-controller
--set vpcId=vpc-***************
--set image.repository= *** /amazon/aws-load-balancer-controller
```

*** https://docs.aws.amazon.com/eks/latest/userguide/add-ons-images.html ***

If you use helm upgrade, you must manually install the CRDs.

```
wget https://raw.githubusercontent.com/aws/eks-charts/master/stable/aws-load-balancer-controller/crds/crds.yaml
kubectl apply -f crds.yaml
```

Step 3: Verify that the controller is installed

```
kubectl get deployment -n kube-system aws-load-balancer-controller
```

EBS

Step 1: Create an IAM Policy for the EBS CSI Driver

The IAM policy defines the permissions required by the EBS CSI driver to interact with AWS resources.

1. Save the following policy in a file named ebs-csi-policy.json:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:AttachVolume",
        "ec2:DetachVolume",
        "ec2:CreateVolume",
        "ec2:DeleteVolume",
        "ec2:ModifyVolume",
        "ec2:DescribeVolumes",
        "ec2:DescribeVolumeAttribute",
        "ec2:DescribeVolumeStatus",
        "ec2:DescribeInstances"
      ],
      "Resource": "*"
    }
  ]
}
```

2. Create the IAM policy:

```
aws iam create-policy --policy-name AmazonEKS_EBS_CSI_Driver_Policy --policy-document file://ebs-csi-policy.json
```

Step 2: Create a Service Account with IAM Role

1. Create the IAM role and associate it with a Kubernetes service account using eksctl:

```
eksctl create iamserviceaccount \
  --name ebs-csi-controller-sa \
  --namespace kube-system \
  --cluster <cluster-name> \
  --attach-policy-arn
arn:aws:iam::<ACCOUNT_ID>:policy/AmazonEKS_EBS_CSI_Driver_Policy \
  --approve \
  --override-existing-serviceaccounts
```

2. Verify the service account:

```
kubectl get serviceaccount ebs-csi-controller-sa -n kube-system
```

Step 3: Install the EBS CSI Driver

1. Add the AWS EBS CSI driver Helm repository:

```
helm repo add aws-ebs-csi-driver https://kubernetes-sigs.github.io/aws-ebs-csi-driver
helm repo update
```

2. Install the driver:

```
helm install aws-ebs-csi-driver aws-ebs-csi-driver/aws-ebs-csi-driver \
  --namespace kube-system \
  --set controller.serviceAccount.create=false \
  --set controller.serviceAccount.name=ebs-csi-controller-sa
```

Step 4: Create a StorageClass

The StorageClass defines how volumes are provisioned dynamically, offering significant advantages over static provisioning. Unlike static provisioning, where administrators must manually create and manage PersistentVolumes (PVs) for each use case, dynamic provisioning allows Kubernetes to automatically provision storage as needed. This reduces manual effort, minimizes errors, and ensures efficient resource utilization, especially in environments with fluctuating storage demands.

1. Apply the following configuration:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ebs-sc
provisioner: ebs.csi.aws.com
parameters:
  type: gp3
  fsType: ext4
  encrypted: "true"
  kmsKeyId: "arn:aws:kms:REGION:ACCOUNT_ID:key/KEY_ID" # Optional, for
encryption
reclaimPolicy: Delete
volumeBindingMode: WaitForFirstConsumer
```

- **Key Parameters:**
- type: Specifies the volume type (e.g., gp2, gp3, io1).
- encrypted: Ensures data is encrypted.
- kmsKeyId: The KMS key used for encryption (optional).
- reclaimPolicy: Controls what happens to the volume when the PVC is deleted (Delete or Retain).
- volumeBindingMode: WaitForFirstConsumer delays provisioning until the pod is scheduled.

Step 5: Create a PersistentVolumeClaim (PVC)

1. Define a PVC using the StorageClass:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: ebs-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: ebs-sc
```

1. Apply the PVC:

```
kubectl apply -f ebs-pvc.yaml
```

Step 6: Deploy a Pod Using the PVC

1. Define a pod that uses the PVC:

```
apiVersion: v1
kind: Pod
metadata:
  name: app-pod
spec:
  containers:
  - name: app-container
    image: nginx
    volumeMounts:
    - mountPath: "/data"
      name: ebs-volume
  volumes:
  - name: ebs-volume
    persistentVolumeClaim:
      claimName: ebs-pvc
```

2. Deploy the pod:

```
kubectl apply -f app-pod.yaml
```

Step 7: Verify the Setup

1. Check the status of the PVC:

```
kubectl get pvc
```

2. Check the pod status and confirm the volume is mounted:

```
kubectl describe pod app-pod
```

3. Verify the EBS volume in the AWS console:

- Navigate to the **Elastic Block Store > Volumes** section.