**01 EKSCTL Important commands**

**Create EKS Node Group in Private Subnets**

* We are going to create a node group in VPC Private Subnets
* We are going to deploy workloads on the private node group wherein workloads will be running private subnets and load balancer gets created in public subnet and accessible via internet.

**Delete existing Public Node Group in EKS Cluster**

**# Get NodeGroups in a EKS Cluster**

--- eksctl get nodegroup --cluster=<Cluster-Name>

--- eksctl get nodegroup --cluster=eksdemo1

**# Delete Node Group - Replace nodegroup name and cluster name**

--- eksctl delete nodegroup <NodeGroup-Name> --cluster <Cluster-Name>

--- eksctl delete nodegroup eksdemo1-ng-public1 --cluster eksdemo1

**Create EKS Node Group in Private Subnets**

* Create Private Node Group in a Cluster
* Key option for the command is --node-private-networking

--- eksctl create nodegroup --cluster=eksdemo1 \

--region=us-east-1 \

--name=eksdemo1-ng-private1 \

--node-type=t3.medium \

--nodes-min=2 \

--nodes-max=4 \

--node-volume-size=20 \

--ssh-access \

--ssh-public-key=kube-demo \

--managed \

--asg-access \

--external-dns-access \

--full-ecr-access \

--appmesh-access \

--alb-ingress-access \

--node-private-networking

**Verify Cluster, Node Groups and configure kubectl cli if not configured**

**# Verify EKS Cluster**

--- eksctl get cluster

**# Verify EKS Node Groups**

--- eksctl get nodegroup --cluster=eksdemo1

**# Verify if any IAM Service Accounts present in EKS Cluster**

--- eksctl get iamserviceaccount --cluster=eksdemo1

**Observation:**

1. No k8s Service accounts as of now.

**# Configure kubeconfig for kubectl**

--- eksctl get cluster # TO GET CLUSTER NAME

--- aws eks --region <region-code> update-kubeconfig --name <cluster\_name>

--- aws eks --region us-east-1 update-kubeconfig --name eksdemo1

**# Verify EKS Nodes in EKS Cluster using kubectl**

--- kubectl get nodes

**# Verify using AWS Management Console**

1. EKS EC2 Nodes (Verify Subnet in Networking Tab)

2. EKS Cluster

**Create IAM Policy**

**## Download specific version**

--- curl -o iam\_policy\_v2.3.1.json https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.3.1/docs/install/iam\_policy.json

**# Create IAM Policy using policy downloaded**

--- aws iam create-policy \

--policy-name AWSLoadBalancerControllerIAMPolicy \

--policy-document <file://iam_policy_latest.json>

**Make a note of Policy ARN**

--- Make a note of Policy ARN as we are going to use that in next step when creating IAM Role.

**# Policy ARN**

Policy ARN: arn:aws:iam::180789647333:policy/AWSLoadBalancerControllerIAMPolicy

**Create an IAM role for the AWS LoadBalancer Controller and attach the role to the Kubernetes service account**

* Applicable only with eksctl managed clusters
* This command will create an AWS IAM role
* This command also will create Kubernetes Service Account in k8s cluster
* In addition, this command will bound IAM Role created and the Kubernetes service account created

**Create IAM Role using eksctl**

**# Verify if any existing service account**

--- kubectl get sa -n kube-system

--- kubectl get sa aws-load-balancer-controller -n kube-system

**Obseravation:**

1. Nothing with name "aws-load-balancer-controller" should exist

**# Template**

eksctl create iamserviceaccount \

--cluster=my\_cluster \

--namespace=kube-system \

--name=aws-load-balancer-controller \ #Note: K8S Service Account Name that need to be bound to newly created IAM Role

--attach-policy-arn=arn:aws:iam::111122223333:policy/AWSLoadBalancerControllerIAMPolicy \

--override-existing-serviceaccounts \

--approve

**# Replaced name, cluster and policy arn (Policy arn we took note in step-02)**

eksctl create iamserviceaccount \

--cluster=eksdemo2 \

--namespace=kube-system \

--name=aws-load-balancer-controller \

--attach-policy-arn=arn:aws:iam::893980973663:policy/AWSLoadBalancerControllerIAMPolicy \

--override-existing-serviceaccounts \

--approve

**Install the AWS Load Balancer Controller using Helm**

**Helm installation**

--- curl https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 > get\_helm.sh

--- chmod 700 get\_helm.sh

--- ./get\_helm.sh

**# Add the eks-charts 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.**

--- helm repo update

**# Install the AWS Load Balancer Controller.**

**## Template**

helm install aws-load-balancer-controller eks/aws-load-balancer-controller \

-n kube-system \

--set clusterName=<cluster-name> \

--set serviceAccount.create=false \

--set serviceAccount.name=aws-load-balancer-controller \

--set region=<region-code> \

--set vpcId=<vpc-xxxxxxxx> \

--set image.repository=<account>.dkr.ecr.<region-code>.amazonaws.com/amazon/aws-load-balancer-controller

**## Replace Cluster Name, Region Code, VPC ID, Image Repo Account ID and Region Code**

helm install aws-load-balancer-controller eks/aws-load-balancer-controller \

-n kube-system \

--set clusterName=eksdemo2 \

--set serviceAccount.create=false \

--set serviceAccount.name=aws-load-balancer-controller \

--set region=us-east-1 \

--set vpcId=vpc-017c915a77ce6dcac\

--set image.repository=8939-8097-3663.dkr.ecr.us-east-1.amazonaws.com/amazon/aws-load-balancer-controller

**Verify that the controller is installed and Webhook Service created**

**# Verify that the controller is installed.**

kubectl -n kube-system get deployment

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

kubectl -n kube-system describe deployment aws-load-balancer-controller

**# Sample Output**

Kalyans-MacBook-Pro:08-01-Load-Balancer-Controller-Install kdaida$ kubectl get deployment -n kube-system aws-load-balancer-controller

NAME READY UP-TO-DATE AVAILABLE AGE

aws-load-balancer-controller 2/2 2 2 27s

Kalyans-MacBook-Pro:08-01-Load-Balancer-Controller-Install kdaida$

**# Verify AWS Load Balancer Controller Webhook service created**

--- kubectl -n kube-system get svc

--- kubectl -n kube-system get svc aws-load-balancer-webhook-service

--- kubectl -n kube-system describe svc aws-load-balancer-webhook-service

**# Sample Output**

Kalyans-MacBook-Pro:aws-eks-kubernetes-masterclass-internal kdaida$ kubectl -n kube-system get svc aws-load-balancer-webhook-service

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

aws-load-balancer-webhook-service ClusterIP 10.100.53.52 <none> 443/TCP 61m

Kalyans-MacBook-Pro:aws-eks-kubernetes-masterclass-internal kdaida$

**# Verify Labels in Service and Selector Labels in Deployment**

--- kubectl -n kube-system get svc aws-load-balancer-webhook-service -o yaml

--- kubectl -n kube-system get deployment aws-load-balancer-controller -o yaml

Observation:

1. Verify "spec.selector" label in "aws-load-balancer-webhook-service"

2. Compare it with "aws-load-balancer-controller" Deployment "spec.selector.matchLabels"

3. Both values should be same which traffic coming to "aws-load-balancer-webhook-service" on port 443 will be sent to port 9443 on "aws-load-balancer-controller" deployment related pods.

**UNINSTALL AWS Load Balancer Controller using Helm Command (Information Purpose - SHOULD NOT EXECUTE THIS COMMAND)**

* This step should not be implemented.
* This is just put it here for us to know how to uninstall aws load balancer controller from EKS Cluster

**# Uninstall AWS Load Balancer Controller**

--- helm uninstall aws-load-balancer-controller -n kube-system

**Create IngressClass Resource**

**# Create IngressClass Resource**

--- kubectl apply -f kube-manifests

**# Verify IngressClass Resource**

--- kubectl get ingressclass

**# Describe IngressClass Resource**

--- kubectl describe ingressclass my-aws-ingress-class