**Creating AWS eks cluster and deploying app-2048**

**Prerequisites**

* kubectl – A command line tool for working with Kubernetes clusters.
* eksctl – A command line tool for working with EKS clusters that automates many individual tasks.
* AWS CLI – A command line tool for working with AWS services, including Amazon EKS.

**Login to AWS console using AWS CLI - Secret key and password**

1. **command**

eksctl create cluster --name demo-cluster --region us-east-1 –fargate

* Command to create eks cluster in AWS console, provide cluster name region
* we can use ec2 instance or fargate, based on the requirements.
* Wait for the control plane ready.
* eks cluster creates public and private subnets within a VPC, in private subnets we create application. These all things will take care by eksctl utility.
* It creates pods, Demon sets, service accounts etc
* **OpenID connect provider URL –** identity provider, ex- okta, authenticator apps**.**

**In case nowadays we see like –** login with Facebook, login with google they all serves as a social identity provider whereas attach it identity provider to Identity broker.

**In AWS case –** AWS allows us to any identity provider like IAM or any other service that you required.

**Here we Integrate IAM identity provider because if this pod wants to talk to S3 bucket or eks control plane or CloudWatch** – easy to talk with any AWS services.

* **Compute -** Fargate Instances are created**.**

**Fargate Profile –** fargate profile attached to default and kube- system namespace, that means we need to deploy pods into these namespaces.

If you want to deploy in any other namespaces, we have create new Fargate.

1. **Command**

**aws eks update-kubeconfig --name demo-cluster -🡪**downloads kubeconfig file

**kubectl get nodes -🡪** shows all pods

1. **Command**

**Create fargate profile –** It creates a new namespace game-2048

eksctl create fargateprofile \

--cluster demo-cluster \

--region us-east-1 \

--name alb-sample-app \

--namespace game-2048

1. **Command**

**Deploy the deployment, service and Ingress**

kubectl apply -f <https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.5.4/docs/examples/2048/2048_full.yaml>

kubectl get pods -n games-208 🡪 shows all pods from namespace game-2048

kubectl get pods -n games-208 -w 🡪 watches all the pods from namespace game-2048

kubectl get svc -n game-2048 🡪 shows the service running on namespace game-2048

kubectl get ingress -n game-2048 🡪 shows the Ingress running on namespace game-2048

* ALB controller (Nothing but Kubernetes pod) which is running need to access to the application load balancer.
* This pod needs to talk to some AWS resources, to talk to AWS resources it needs to have IAM Integrator. So, we need to create IAM OIDC provider.

eksctl utils associate-iam-oidc-provider --cluster $demo-cluster –approve

**Now, Create ALB Ingress controller (Pod) 🡪 need to access AWS services such as ALB**

**Download IAM policy**

curl -O <https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.5.4/docs/install/iam_policy.json>

**Create IAM Policy**

aws iam create-policy --policy-name AWSLoadBalancerControllerIAMPolicy --policy-document <file://iam_policy.json>

**Create IAM Role -** Attaching Role to the service account of the pod

* **Whenever the pod is running the pod will have a service account, for that service account role must attach so that pod can interact with AWS services.**

eksctl create iamserviceaccount --cluster=<your-cluster-name> --namespace=kube-system --name=aws-load-balancer-controller --role-name AmazonEKSLoadBalancerControllerRole --attach-policy-arn=arn:aws:iam::<your-aws-account-id>:policy/AWSLoadBalancerControllerIAMPolicy –approve

* **For creating ALB, we use Helm Chart**
* **Helm chart creates actual controller, this uses the service account for running the pod**

helm repo add eks <https://aws.github.io/eks-charts> --> **Add helm repo**

helm repo update eks 🡪 **Update the Repo**

**Install the helm chart**

helm install aws-load-balancer-controller eks/aws-load-balancer-controller -n kube-system \

--set clusterName=<your-cluster-name> \

--set serviceAccount.create=false \

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

--set region=<region> \

--set vpcId=<your-vpc-id>

* **Now check load balancer in EC2 service.**
* **With DNS Name URL, we can access application**.

Few commands:

**kubectl get deployment -n kube-system aws-load-balancer-controller** 🡪 Verify that the deployments are running or not

**Kubectl get deploy -n kube-system 🡪** lists all Deployments in the kube-system namespace**.**

**Kubectl edit deploy/aws-load-balancer-controller -n kube-system –** we can check logs of aws-load-balance-controller

