EKS provides a highly available Control Plane . Everything related to Control Plane is handled by EKS. We cant be able to know where the EKS control plane servers are running.

We can attach the worker nodes to EKS through two ways:

1. By using EC2 machines
2. By Using fargates. Fargates provide AWS Serverless compute machines that allows us to run containers. If we use fargate then the whole cluster becomes automatically managed. Worker nodes managed by fargate and Control plane managed by EKS.

Using fargate and EKS we can build a robust and highly stable EKS cluster.

Normal problems with cluster:

Certificate expired

API Server slowness

ETCD crashed

Scheduler not working

Tools to be installed in the machine to interact with EKS:

Kubectl and eksctl

<https://docs.aws.amazon.com/emr/latest/EMR-on-EKS-DevelopmentGuide/setting-up-eksctl.html>

First Install Choocolatey:

<https://chocolatey.org/install>

Login to powershell with admin access:

A blue screen with yellow text

Description automatically generated

A screenshot of a computer

Description automatically generated

A screen shot of a computer code

Description automatically generated

A screenshot of a computer program

Description automatically generated

Chocolatey is ready.

Install eksctl:

<https://docs.aws.amazon.com/emr/latest/EMR-on-EKS-DevelopmentGuide/setting-up-eksctl.html>

A screenshot of a computer

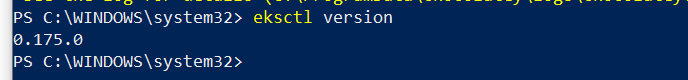
Description automatically generated

A screenshot of a computer

Description automatically generated

A screen shot of a computer program

Description automatically generated



Run the above commands in powershell.

Create EKS cluster:

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

*The above commands create the following resources:*

*VPC, 2 public subnets, 2 private subnets, Internet gateway, 2 private route table, 1 public route table, NAT gateways, Elastic IP, Securiy Groups, EKS Cluster, fargate profiles.*

A screenshot of a computer

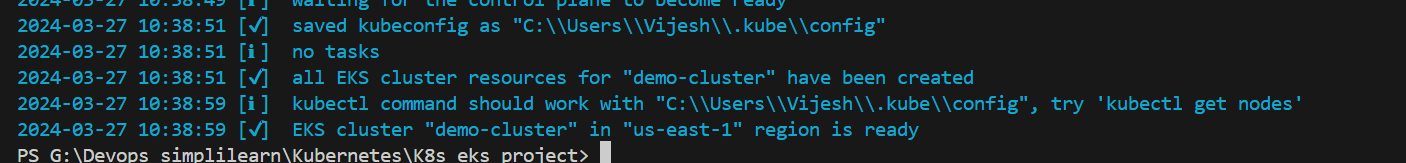
Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated



After creating eks cluster we need to create a fargate profile which can be used to deploy resources like pods, deployments in a different namespace. By default the fargate profile allows to create resources in default/kube-system namespaces.

A screenshot of a computer

Description automatically generated

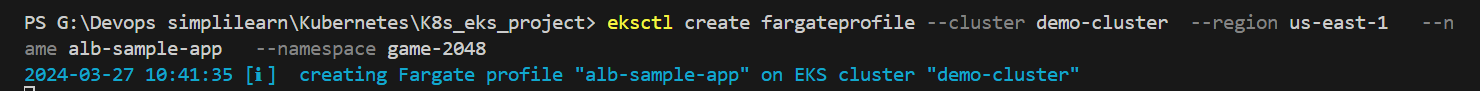
eksctl create fargateprofile \

--cluster demo-cluster \

--region us-east-1 \

--name alb-sample-app \

--namespace game-2048



eksctl create fargateprofile --cluster demo-cluster --region us-east-1 --name alb-sample-app --namespace game-2048

A screenshot of a computer

Description automatically generated

**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

A black screen with colorful text

Description automatically generated

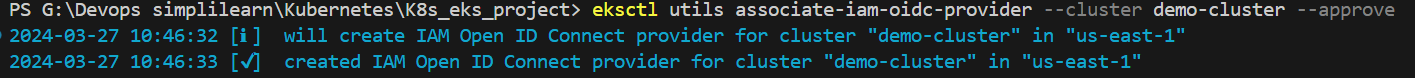
A computer screen with white text

Description automatically generated

For ingress to get the address , create an ingress controller:

**Commands to configure IAM OIDC provider**

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



**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**

eksctl create iamserviceaccount \

--cluster=demo-cluster \

--namespace=kube-system \

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

--role-name AmazonEKSLoadBalancerControllerRole \

--attach-policy-arn=arn:aws:iam::464598805321:policy/AWSLoadBalancerControllerIAMPolicy \

--approve

A computer screen shot of a black screen

Description automatically generated

A computer screen with text

Description automatically generated

**Deploy ALB controller**

Install helm if not there to deploy ALB using helm charts.

**Install Helm in Linux**

**$** curl -fsSL -o get\_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3

**$** chmod 700 get\_helm.sh

**$** ./get\_helm.sh

ubuntu@ip-172-31-31-5:~$ helm version

version.BuildInfo{Version:"v3.14.3", GitCommit:"f03cc04caaa8f6d7c3e67cf918929150cf6f3f12", GitTreeState:"clean", GoVersion:"go1.21.7"}

**In windows helm install:**

Go to powershell and run as admin :

choco install kubernetes-helm

A computer screen with text on it

Description automatically generated

A screen shot of a computer code

Description automatically generated

**Add helm repo**

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

**Update the repo**

helm repo update eks

**Install**

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>

or

helm install aws-load-balancer-controller eks/aws-load-balancer-controller -n kube-system --set clusterName=demo-cluster --set serviceAccount.create=false --set serviceAccount.name=aws-load-balancer-controller --set region=us-east-1 --set vpcId=vpc-098e0b4530595d35f

A black screen with yellow and white text

Description automatically generated

**Verify that the deployments are running.**

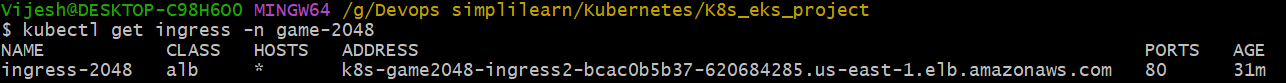
kubectl get deployment -n kube-system aws-load-balancer-controller

ALB controller are installed in kube-system namespaces:

A computer screen with white text and numbers

Description automatically generated

Now ingress resource should be getting the address



A screenshot of a computer

Description automatically generated

A screenshot of a computer

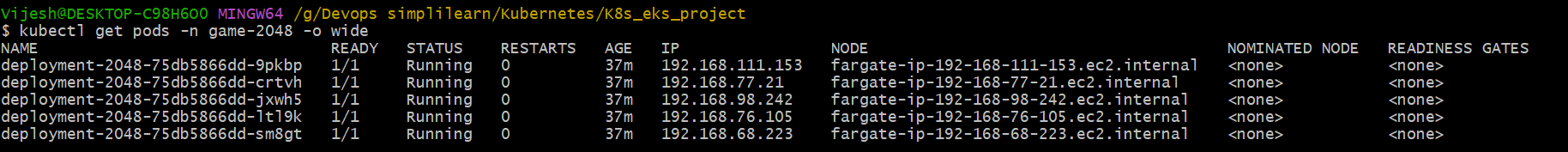
Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated



The above load balancer dns address is assigned to ingress resource. Now browse through the ingress resource address to check the application:

A screenshot of a game

Description automatically generated

Whatever resources we create using eksctl there will be a cloudformation stack:

A screenshot of a computer

Description automatically generated

**To delete the eks cluster:**

eksctl delete cluster --name demo-cluster --region us-east-1

----------------------------------------Completed ---------------------------------------------------------------------------