AWS Elastic Kubernetes Service

YouTube:
https://www.youtube.com/watch?v=RRCrY12VY_s&list=PLdpzxOOAlwvJdsW6A0jCz_3VaANuFMLpc&index=16

If we are not using any EKS, AKS or any other then we need to spin up servers for each master and worker nodes.

Kubernetes will have Control Plane (Master Node) and Data Plane (Worker Node). If our organisation needs 3 master 3 mode architecture for K8s cluster then we need to spin up 6 EC2 instances and then install all the components of Control plane like API, ETCD, Scheduler, CCM and we have to install Container runtime, DNS, Kube-Proxy and then we need to configure worker nodes with master node.

It will be our task to do. If we choose EKS then we no need to worry about creating cluster.

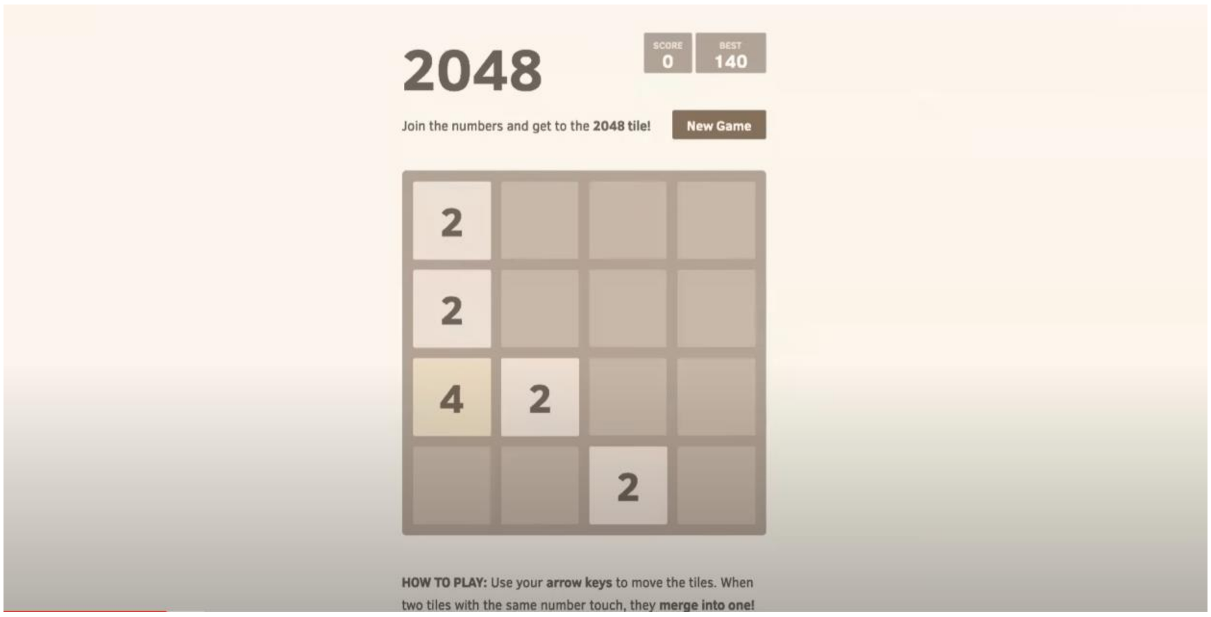
So, AWS is providing a solution to this with EKS. EKS is a Managed Control plane.

EKS is a managed Kubernetes offering that simplifies the setup of a high available Kubernetes cluster on AWS.

- EKS eliminates the need for manually provisioning and managing control plane components such as API server, etcd, scheduler, and controller manager.
- Users can easily access applications deployed on worker nodes by interacting with the control plane.

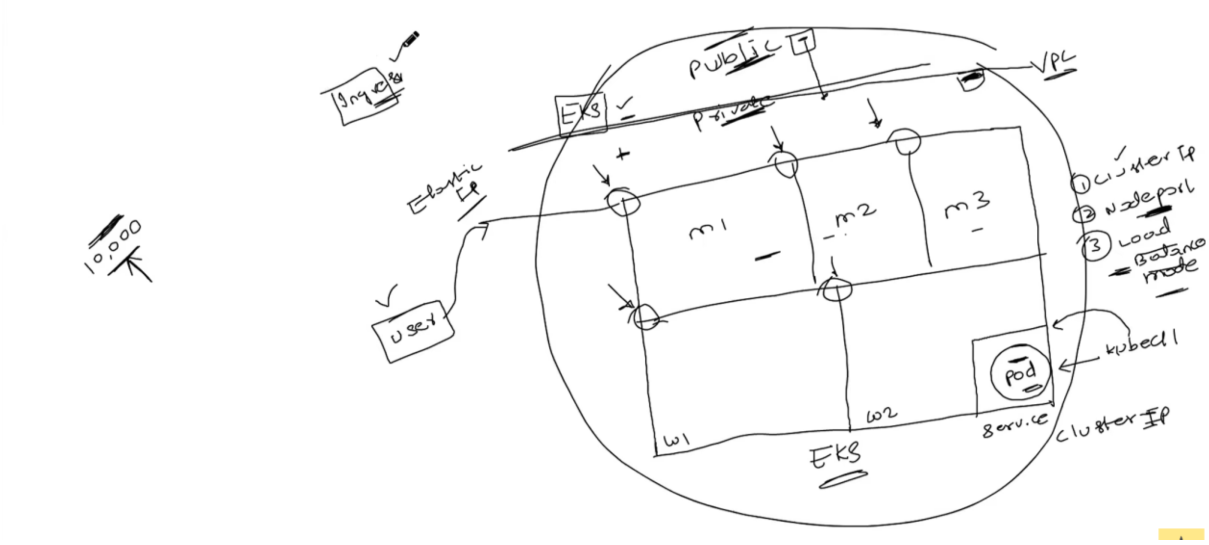
When we are creating cluster using EKS, it creates a high available Control plane in Master nodes and allows easily to add worker nodes.

We can add EC2 instances as worker nodes or we can use **Fargate**, an AWS Serverless Compute which runs as a container. But if we are choosing EC2 instances for worker nodes then we need make the nodes as High available, configure them with Master. Basically, we need to manage them. If we are using Fargate then we no need to worry at all.

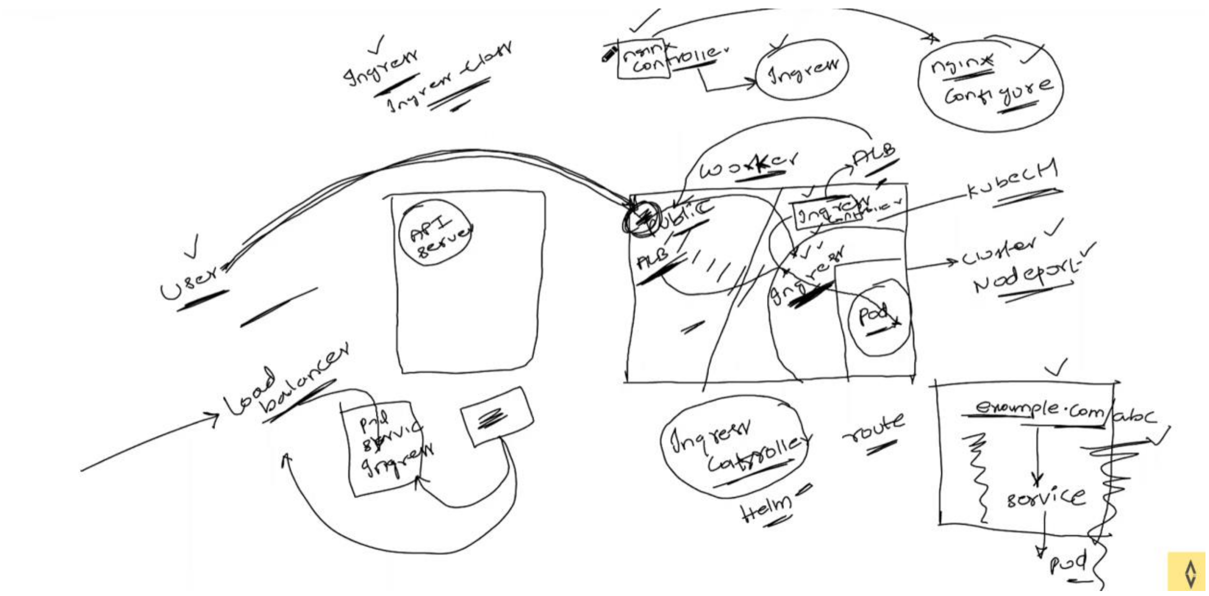

2048
SCORE 0
BEST 140
Join the numbers and get to the 2048 tile!
New Game
HOW TO PLAY: Use your arrow keys to move the tiles. When two tiles with the same number touch, they merge into one!

We are going to deploy this application into our EKS Cluster.

Say, we have created 3 master and 2 worker nodes. We have our application in a pod in w2 node. We will create a pod and for that pod we will create service. Ususally they will have ClusterIP as a service but we will also have NodePort and Load Balancer types of service as well. The whole cluster will be inside a Private VPC. The user should be able to access the application inside the Pod from external world. If you use Cluster IP mode allows access within the cluster only. Node Port mode allows access from any IP address within the cluster, but limited to the private subnet. But all the resources in the cluster should be assigned with Elastic IP. This will be very expensive if we have 1000s of Pods.

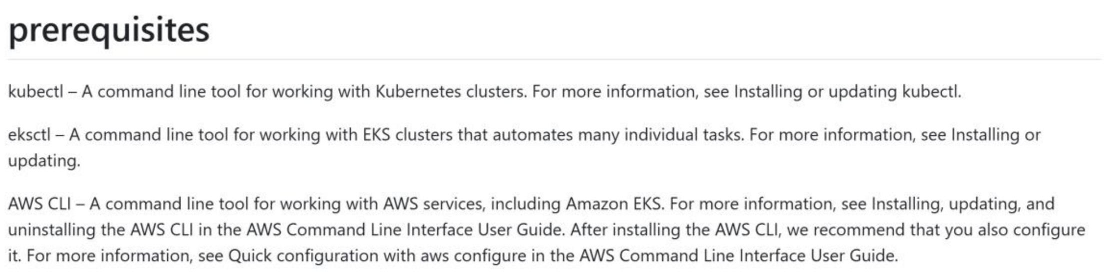


That's why we need to use Ingress. Ingress will be created on top of Service. Ingress Controller will look for Ingress getting requests. Ingress Controller will create Application load balancer which the user will access and it gets redirect to the Ingress and then to the Service and then to the Pod.



We had lot of Ingress controllers in the market we choose any one of them. Example: NginX

Repo: https://github.com/iam-veeramalla/aws-devops-zero-to-hero/tree/main/day-22



prerequisites

kubectl – A command line tool for working with Kubernetes clusters. For more information, see Installing or updating kubectl.

eksctl – A command line tool for working with EKS clusters that automates many individual tasks. For more information, see Installing or updating.

AWS CLI – A command line tool for working with AWS services, including Amazon EKS. For more information, see Installing, updating, and uninstalling the AWS CLI in the AWS Command Line Interface User Guide. After installing the AWS CLI, we recommend that you also configure it. For more information, see Quick configuration with aws configure in the AWS Command Line Interface User Guide.

Take help from official documentations for installing these. EKSCTL is installed using ChatGPT and i-am-authentication is also required. Which is also installed with help of ChatGPT. Refer to GitHub Release pages for both.

We can create Cluster using EKS service in AWS but we need to provide a lot of information while creating it. But also, we can create using EKSCTL (Command line utility that is used to manage EKS Cluster).

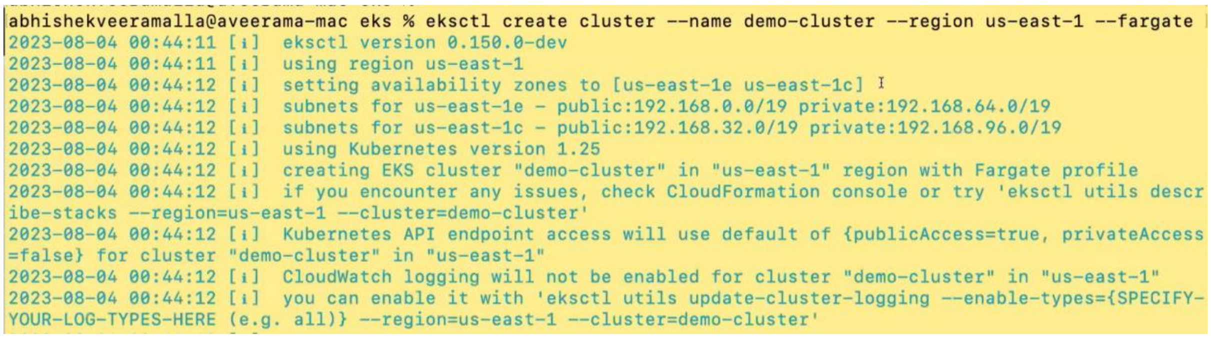
For this project we are directly using Root account of AWS because if we want to user a separate user then we need to create IAM roles and add lot of permissions.

Let's create Cluster using EKSCTL. It is useful for Create, get, list and delete clusters

- Create, drain and delete node groups
- Scale a node group
- Update a cluster
- Use custom AMIs
- Configure VPC Networking
- Configure access to API endpoints
- Support for GPU nodegroups
- Spot instances and mixed instances
- IAM Management and Add-on Policies
- List cluster CloudFormation stacks
- Install coredns
- Write kubeconfig file for a cluster

CMD to Create EKS Cluster: eksctl create cluster --name demo-cluster --region us-east-1 –fargate

We are using fargate for worker nodes.

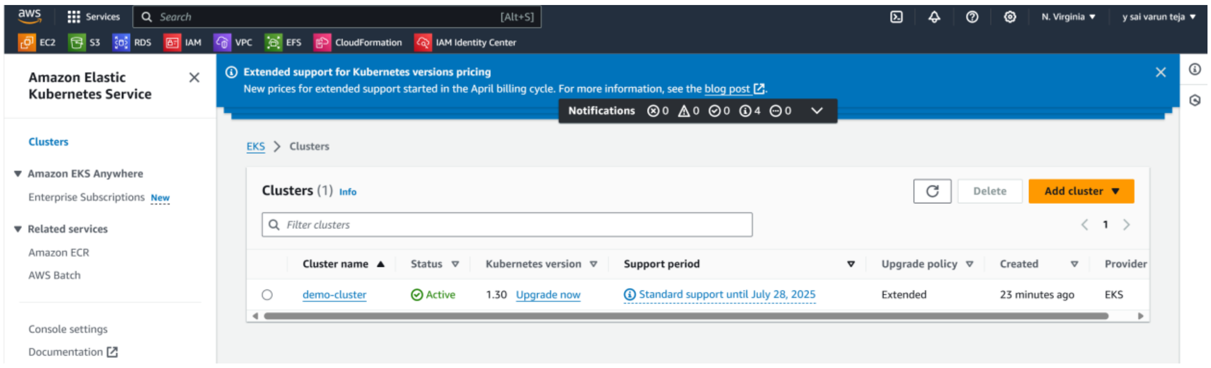


```
abhishekveeramalla@aveerama-mac eks % eksctl create cluster --name demo-cluster --region us-east-1 --fargate
2023-08-04 00:44:11 [ℹ]  eksctl version 0.150.0-dev
2023-08-04 00:44:11 [ℹ]  using region us-east-1
2023-08-04 00:44:12 [ℹ]  setting availability zones to [us-east-1e us-east-1c]
2023-08-04 00:44:12 [ℹ]  subnets for us-east-1e - public:192.168.0.0/19 private:192.168.64.0/19
2023-08-04 00:44:12 [ℹ]  subnets for us-east-1c - public:192.168.32.0/19 private:192.168.96.0/19
2023-08-04 00:44:12 [ℹ]  using Kubernetes version 1.25
2023-08-04 00:44:12 [ℹ]  creating EKS cluster "demo-cluster" in "us-east-1" region with Fargate profile
2023-08-04 00:44:12 [ℹ]  if you encounter any issues, check CloudFormation console or try 'eksctl utils descr
ibe-stacks --region=us-east-1 --cluster=demo-cluster'
2023-08-04 00:44:12 [ℹ]  Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess
=false} for cluster "demo-cluster" in "us-east-1"
2023-08-04 00:44:12 [ℹ]  CloudWatch logging will not be enabled for cluster "demo-cluster" in "us-east-1"
2023-08-04 00:44:12 [ℹ]  you can enable it with 'eksctl utils update-cluster-logging --enable-types={SPECIFY-
YOUR-LOG-TYPES-HERE (e.g. all)} --region=us-east-1 --cluster=demo-cluster'
```

It creates everuthing for us that is required to create EKS using AWS console. It creates public and private subnets and the application will be inside th private subnets.

It will 15 – 20 mins for creating EKS Cluster. Wait for control plane to get ready.

Switch to us-east-1 region in AWS console:



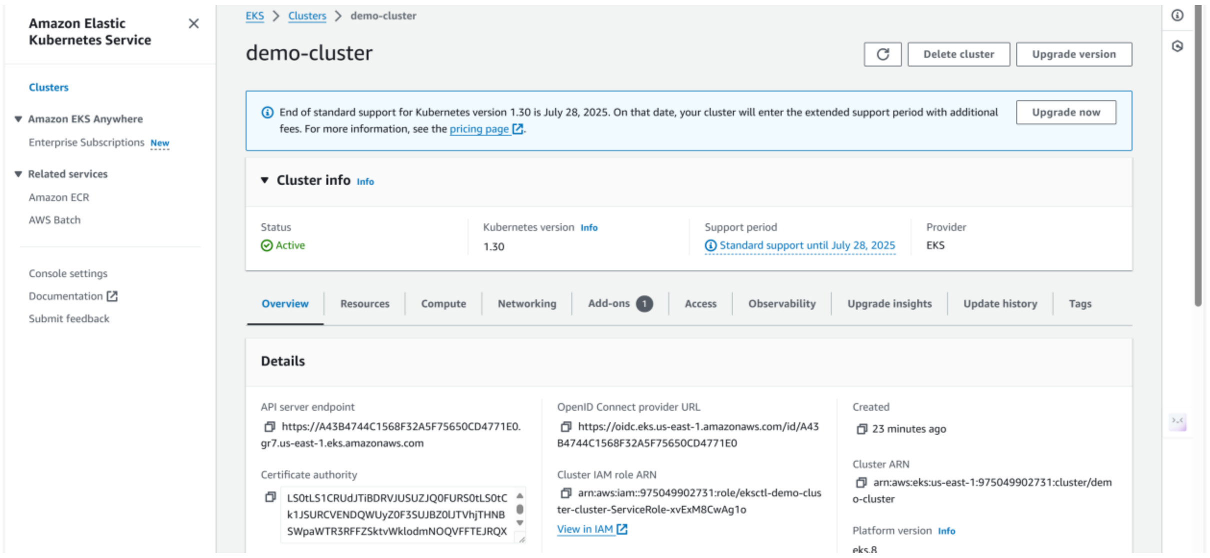
Extended support for Kubernetes versions pricing
New prices for extended support started in the April billing cycle. For more information, see the blog post.

Notifications ⊗0 ⚠0 ⊘0 ⓘ4 ⊖0

Clusters (1) Info

	Cluster name	Status	Kubernetes version	Support period	Upgrade policy	Created	Provider
○	demo-cluster	✓ Active	1.30 Upgrade now	Standard support until July 28, 2025	Extended	23 minutes ago	EKS

Open the cluster, you can see the all the details about the cluster.



demo-cluster

End of standard support for Kubernetes version 1.30 is July 28, 2025. On that date, your cluster will enter the extended support period with additional fees. For more information, see the pricing page.

Cluster info Info

Status	Kubernetes version Info	Support period	Provider
✓ Active	1.30	Standard support until July 28, 2025	EKS

Details

API server endpoint
https://A43B4744C1568F32A5F75650CD4771E0.gr7.us-east-1.eks.amazonaws.com

Certificate authority
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tC
k1JSURCVENDQWUyZ0F3SUJBZ0lJTVhjTHNB
SWpaWTR3RFFZSktvWklodmNOQVFFTEJRQX

OpenID Connect provider URL
https://oidc.eks.us-east-1.amazonaws.com/id/A43B4744C1568F32A5F75650CD4771E0

Cluster IAM role ARN
arn:aws:iam::975049902731:role/eksctl-demo-cluster-cluster-ServiceRole-xvExM8CwAg1o
View in IAM

Created
23 minutes ago

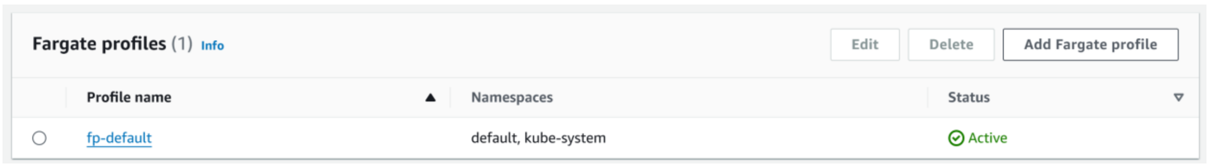
Cluster ARN
arn:aws:eks:us-east-1:975049902731:cluster/demo-cluster

Platform version Info
eks.8

We can check the resources available in our cluster. We no need to give any commands in the CLI.

We have to integrate an authentication method to our Cluster. We can assign IAM or any other as well. Whenever an resource in the cluster wants to talk with resource in the EKS we need to have an authentication. To K8s Pods we can attach or integrate IAM rules with k8s service accounts. So that, we can talk to any other AWS Services.

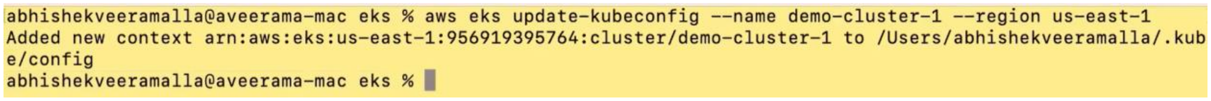
Under Compute section we can see Fargate profile



Fargate profiles (1) Info

	Profile name	Namespaces	Status
○	fp-default	default, kube-system	✓ Active

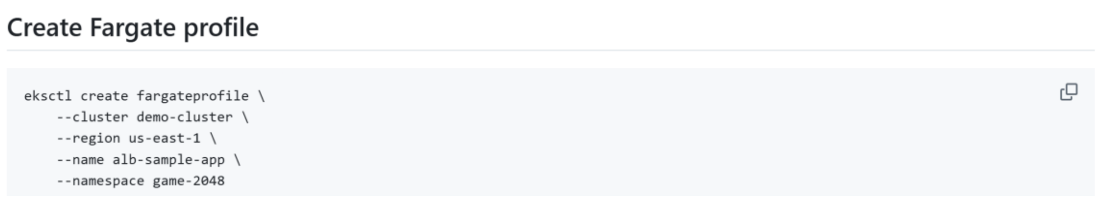
This is very imporrtant. If we want to deploy pods to any namespace it is not possible. We can deploy to default or kube-system name spaces only. If we wnt to deploy to different namespace then we have to create a new fargate profile.



```
abhishekveeramalla@aveerama-mac eks % aws eks update-kubeconfig --name demo-cluster-1 --region us-east-1
Added new context arn:aws:eks:us-east-1:956919395764:cluster/demo-cluster-1 to /Users/abhishekveeramalla/.kub
e/config
abhishekveeramalla@aveerama-mac eks % 
```

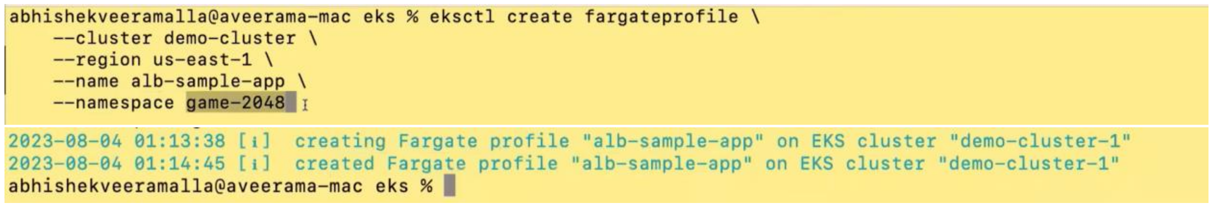
Let’s try to deploy the actual 2048 application.

URL: https://github.com/iam-veeramalla/aws-devops-zero-to-hero/blob/main/day-22/2048-app-deploy-ingress.md


Create Fargate profile

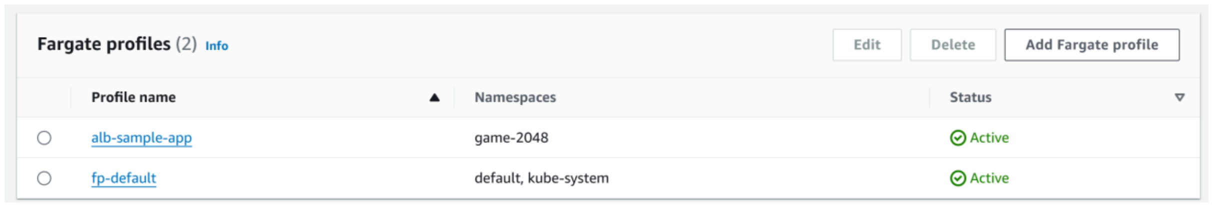
```
eksctl create fargateprofile \
    --cluster demo-cluster \
    --region us-east-1 \
    --name alb-sample-app \
    --namespace game-2048
```

We are adding a namespace for fargate as well here.



```
abhishekveeramalla@aveerama-mac eks % eksctl create fargateprofile \
    --cluster demo-cluster \
    --region us-east-1 \
    --name alb-sample-app \
    --namespace game-2048
2023-08-04 01:13:38 [ℹ]  creating Fargate profile "alb-sample-app" on EKS cluster "demo-cluster-1"
2023-08-04 01:14:45 [ℹ]  created Fargate profile "alb-sample-app" on EKS cluster "demo-cluster-1"
abhishekveeramalla@aveerama-mac eks %
```

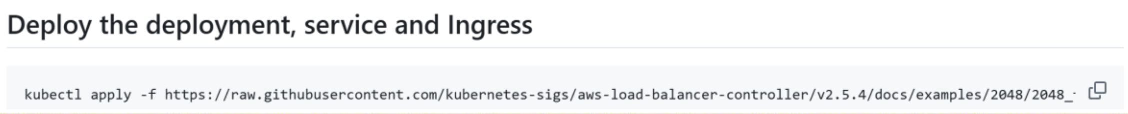
New fargate is created with new namespace.



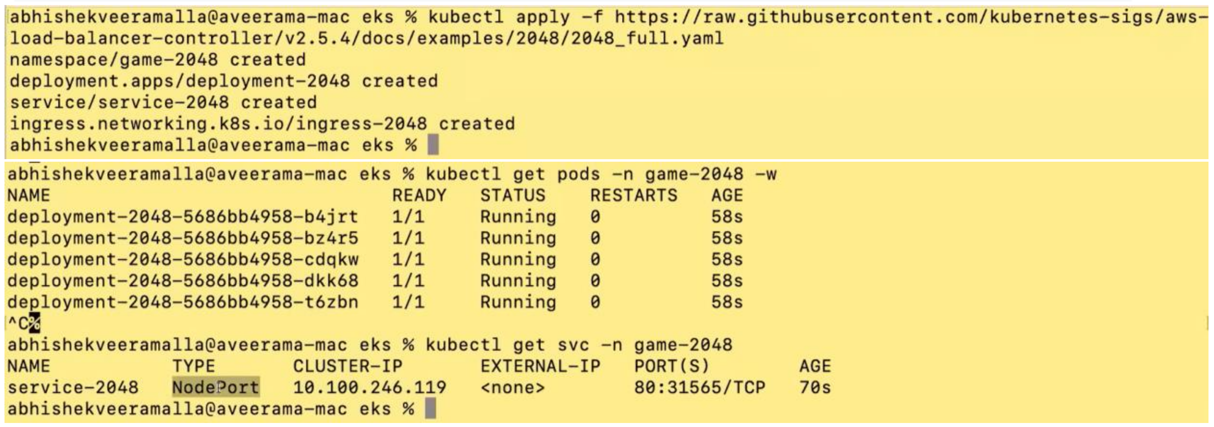
Fargate profiles (2) Info

	Profile name	Namespaces	Status
○	alb-sample-app	game-2048	✓ Active
○	fp-default	default, kube-system	✓ Active

Now,


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_
```

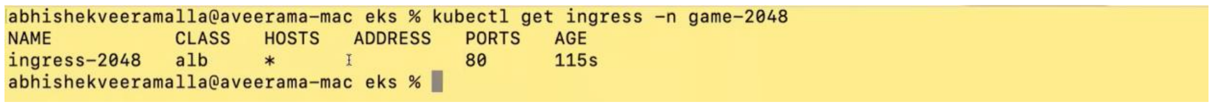


```
abhishekveeramalla@aveerama-mac eks % kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-
load-balancer-controller/v2.5.4/docs/examples/2048/2048_full.yaml
namespace/game-2048 created
deployment.apps/deployment-2048 created
service/service-2048 created
ingress.networking.k8s.io/ingress-2048 created
abhishekveeramalla@aveerama-mac eks %
```

```
abhishekveeramalla@aveerama-mac eks % kubectl get pods -n game-2048 -w
NAME                               READY   STATUS    RESTARTS   AGE
deployment-2048-5686bb4958-b4jrt   1/1     Running   0          58s
deployment-2048-5686bb4958-bz4r5   1/1     Running   0          58s
deployment-2048-5686bb4958-cdqkw   1/1     Running   0          58s
deployment-2048-5686bb4958-dkk68   1/1     Running   0          58s
deployment-2048-5686bb4958-t6zbn   1/1     Running   0          58s
^C%
abhishekveeramalla@aveerama-mac eks % kubectl get svc -n game-2048
NAME           TYPE       CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
service-2048   NodePort   10.100.246.119   <none>        80:31565/TCP   70s
abhishekveeramalla@aveerama-mac eks %
```

We don't have the External IP. Means anyone within the AWS VPC or who had access to the VPC can talk to this pod using the Node IP address following by Port number.

Our goal is to make this app available to the outside world. For that we have to create an Ingress. Ingress is already created.



```
abhishekveeramalla@aveerama-mac eks % kubectl get ingress -n game-2048
NAME           CLASS   HOSTS   ADDRESS   PORTS   AGE
ingress-2048   alb     *                 80      115s
abhishekveeramalla@aveerama-mac eks %
```

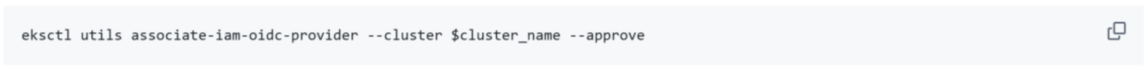
There is no address here for ingress. We need to create an Ingress controller also. Then we will get that address. We have to use this address to access the application form the outside world.

We will create an Ingress controller now, then it will read the Ingress resource called ingress-2048 and then it will create Load balancer.

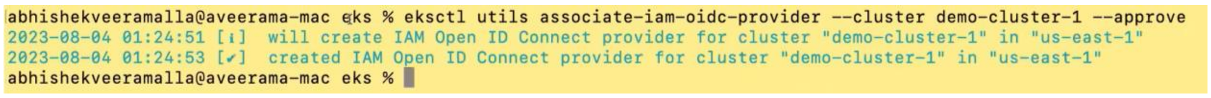
We have to deploy the Ingress controller called as Application Load balancer.

Alb Controller (nothing but a K8s pod) needs to access/talk the application load balancer. So, we need to configure IAM OIDC provider.

URL: https://github.com/iam-veeramalla/aws-devops-zero-to-hero/blob/main/day-22/configure-oidc-connector.md



```
eksctl utils associate-iam-oidc-provider --cluster $cluster_name --approve
```

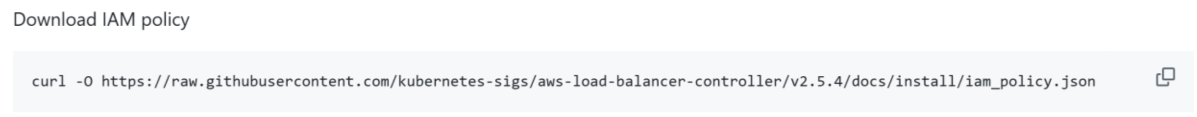


```
abhishekveeramalla@aveerama-mac eks % eksctl utils associate-iam-oidc-provider --cluster demo-cluster-1 --approve
2023-08-04 01:24:51 [ℹ]  will create IAM Open ID Connect provider for cluster "demo-cluster-1" in "us-east-1"
2023-08-04 01:24:53 [✔]  created IAM Open ID Connect provider for cluster "demo-cluster-1" in "us-east-1"
abhishekveeramalla@aveerama-mac eks %
```

If it is asking for region then add --region us-east-1

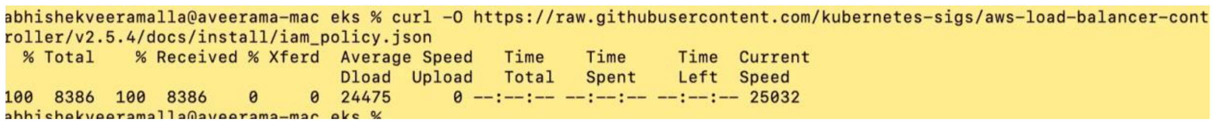
ALB Controller will be created as a Pod and it should have access to AWS services such as ALB.

URL: https://github.com/iam-veeramalla/aws-devops-zero-to-hero/blob/main/day-22/alb-controller-add-on.md

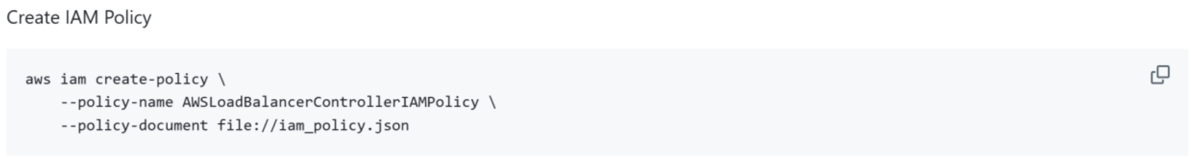


Download IAM policy

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

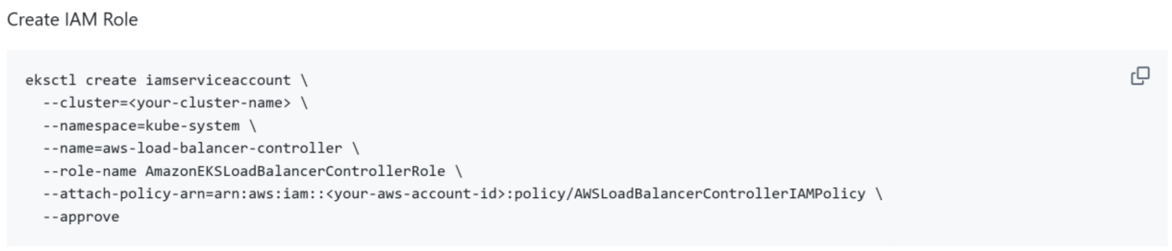


```
abhishekveeramalla@aveerama-mac eks % curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-cont
roller/v2.5.4/docs/install/iam_policy.json
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  8386  100  8386    0     0  24475      0 --:--:-- --:--:-- --:--:-- 25032
abhishekveeramalla@aveerama-mac eks %
```



Create IAM Policy

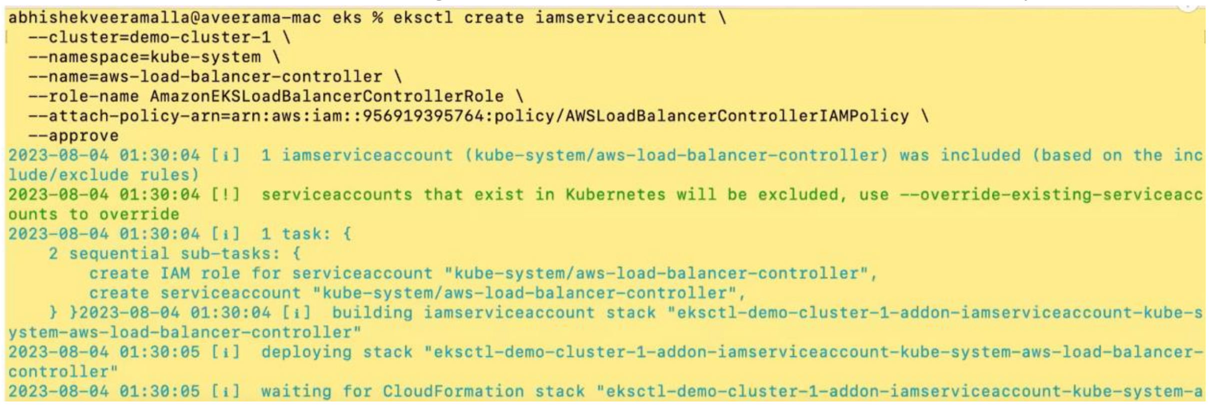
```
aws iam create-policy \
    --policy-name AWSLoadBalancerControllerIAMPolicy \
    --policy-document file://iam_policy.json
```



Create IAM Role

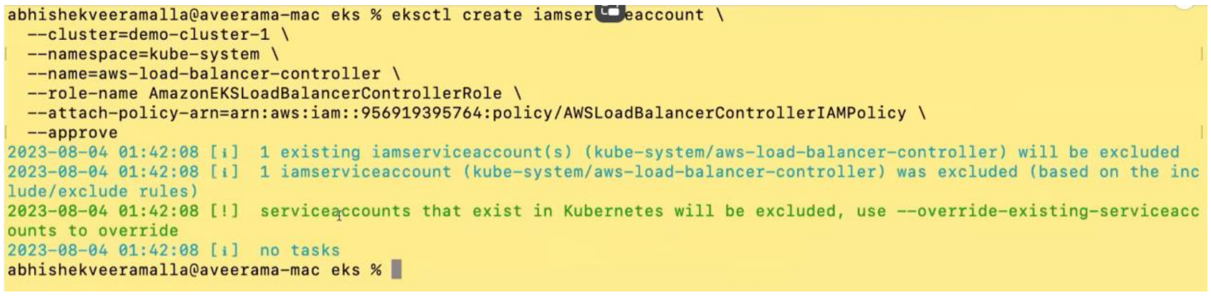
```
eksctl create iamserviceaccount \
  --cluster=<your-cluster-name> \
  --namespace=kube-system \
  --name=aws-load-balancer-controller \
  --role-name AmazonEKSLoadBalancerControllerRole \
  --attach-policy-arn=arn:aws:iam::<your-aws-account-id>:policy/AWSLoadBalancerControllerIAMPolicy \
  --approve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

Execute both commands. Then change the cluster name and aws-account ID in above pic.



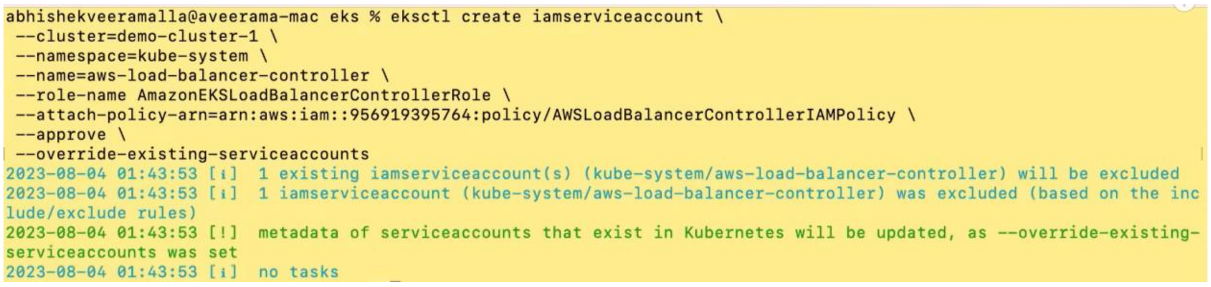
```
abhishekveeramalla@aveerama-mac eks % eksctl create iamserviceaccount \
  --cluster=demo-cluster-1 \
  --namespace=kube-system \
  --name=aws-load-balancer-controller \
  --role-name AmazonEKSLoadBalancerControllerRole \
  --attach-policy-arn=arn:aws:iam::956919395764:policy/AWSLoadBalancerControllerIAMPolicy \
  --approve
2023-08-04 01:30:04 [ℹ]  1 iamserviceaccount (kube-system/aws-load-balancer-controller) was included (based on the inc
lude/exclude rules)
2023-08-04 01:30:04 [!]  serviceaccounts that exist in Kubernetes will be excluded, use --override-existing-serviceacc
ounts to override
2023-08-04 01:30:04 [ℹ]  1 task: {
    2 sequential sub-tasks: {
        create IAM role for serviceaccount "kube-system/aws-load-balancer-controller",
        create serviceaccount "kube-system/aws-load-balancer-controller",
    } }2023-08-04 01:30:04 [ℹ]  building iamserviceaccount stack "eksctl-demo-cluster-1-addon-iamserviceaccount-kube-s
ystem-aws-load-balancer-controller"
2023-08-04 01:30:05 [ℹ]  deploying stack "eksctl-demo-cluster-1-addon-iamserviceaccount-kube-system-aws-load-balancer-
controller"
2023-08-04 01:30:05 [ℹ]  waiting for CloudFormation stack "eksctl-demo-cluster-1-addon-iamserviceaccount-kube-system-a
```

If it is failed, try again.

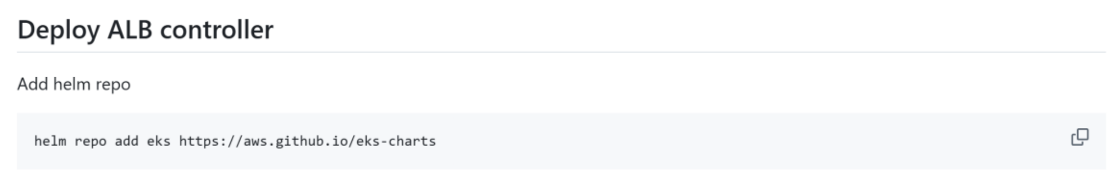


```
abhishekveeramalla@aveerama-mac eks % eksctl create iamser  eaccount \
  --cluster=demo-cluster-1 \
  --namespace=kube-system \
  --name=aws-load-balancer-controller \
  --role-name AmazonEKSLoadBalancerControllerRole \
  --attach-policy-arn=arn:aws:iam::956919395764:policy/AWSLoadBalancerControllerIAMPolicy \
  --approve
2023-08-04 01:42:08 [ℹ]  1 existing iamserviceaccount(s) (kube-system/aws-load-balancer-controller) will be excluded
2023-08-04 01:42:08 [ℹ]  1 iamserviceaccount (kube-system/aws-load-balancer-controller) was excluded (based on the inc
lude/exclude rules)
2023-08-04 01:42:08 [!]  serviceaccounts that exist in Kubernetes will be excluded, use --override-existing-serviceacc
ounts to override
2023-08-04 01:42:08 [ℹ]  no tasks
abhishekveeramalla@aveerama-mac eks %
```

We have to override now.



```
abhishekveeramalla@aveerama-mac eks % eksctl create iamserviceaccount \
 --cluster=demo-cluster-1 \
 --namespace=kube-system \
 --name=aws-load-balancer-controller \
 --role-name AmazonEKSLoadBalancerControllerRole \
 --attach-policy-arn=arn:aws:iam::956919395764:policy/AWSLoadBalancerControllerIAMPolicy \
 --approve \
 --override-existing-serviceaccounts
2023-08-04 01:43:53 [i]  1 existing iamserviceaccount(s) (kube-system/aws-load-balancer-controller) will be excluded
2023-08-04 01:43:53 [i]  1 iamserviceaccount (kube-system/aws-load-balancer-controller) was excluded (based on the inc
lude/exclude rules)
2023-08-04 01:43:53 [!]  metadata of serviceaccounts that exist in Kubernetes will be updated, as --override-existing-
serviceaccounts was set
2023-08-04 01:43:53 [i]  no tasks
```

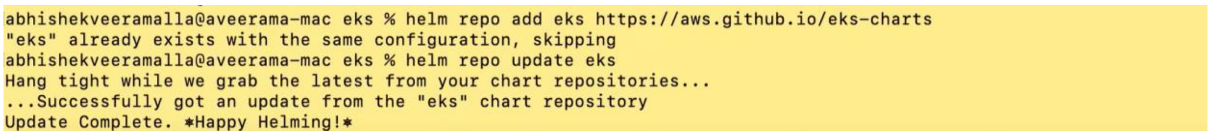


Deploy ALB controller

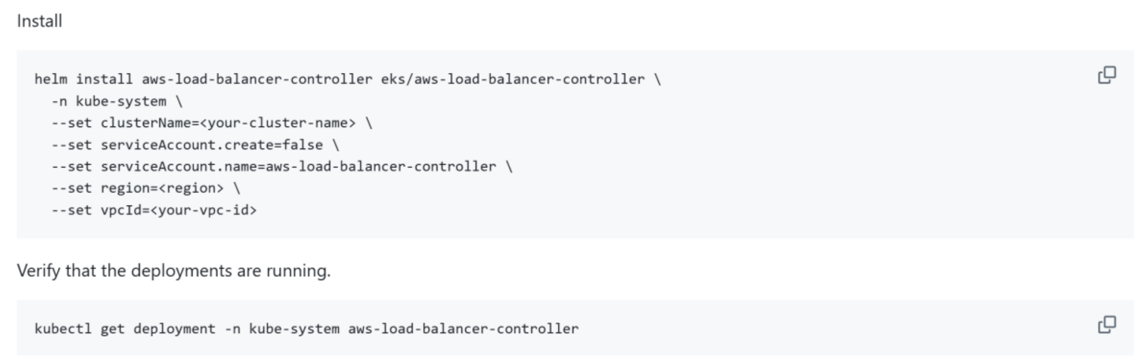
Add helm repo

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

Pre-Requsite: Helm in PC.



```
abhishekveeramalla@aveerama-mac eks % helm repo add eks https://aws.github.io/eks-charts
"eks" already exists with the same configuration, skipping
abhishekveeramalla@aveerama-mac eks % helm repo update eks
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "eks" chart repository
Update Complete. *Happy Helming!*
```



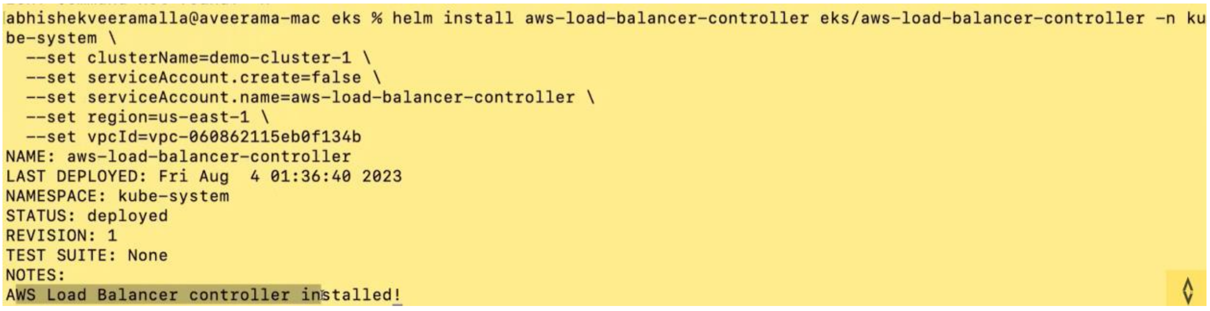
Install

```
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>
```

Verify that the deployments are running.

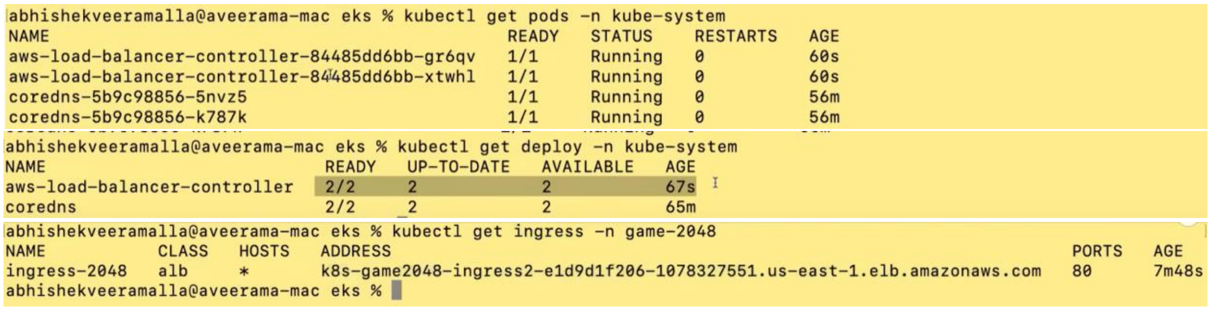
```
kubectl get deployment -n kube-system aws-load-balancer-controller
```

Give the VPC Id, Region, Cluster name.



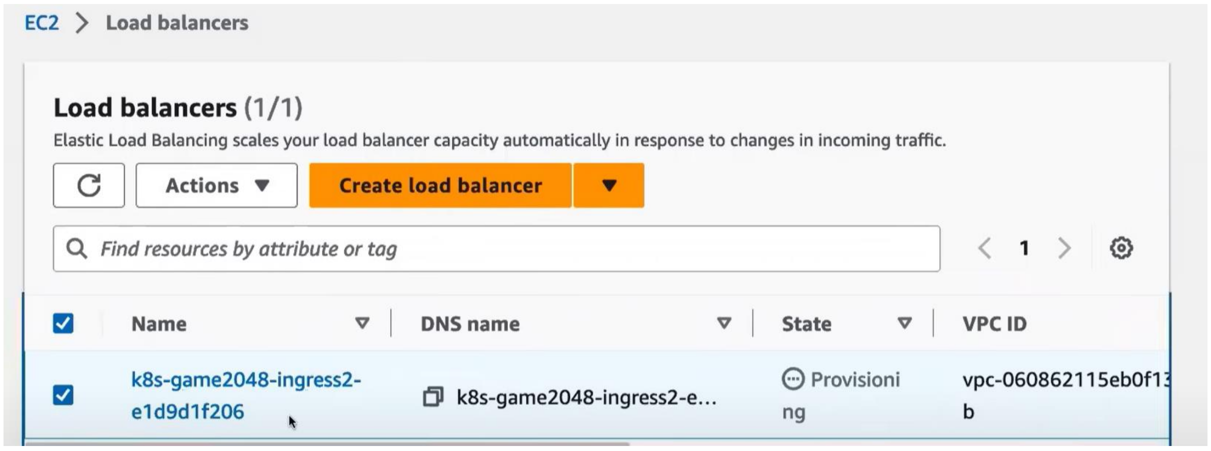
```
abhishekveeramalla@aveerama-mac eks % helm install aws-load-balancer-controller eks/aws-load-balancer-controller -n ku
be-system \
  --set clusterName=demo-cluster-1 \
  --set serviceAccount.create=false \
  --set serviceAccount.name=aws-load-balancer-controller \
  --set region=us-east-1 \
  --set vpcId=vpc-060862115eb0f134b
NAME: aws-load-balancer-controller
LAST DEPLOYED: Fri Aug  4 01:36:40 2023
NAMESPACE: kube-system
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
AWS Load Balancer controller installed!
```

Check the white spaces here for this command. Application load balancer is installed.



```
abhishekveeramalla@aveerama-mac eks % kubectl get pods -n kube-system
NAME                                            READY   STATUS    RESTARTS   AGE
aws-load-balancer-controller-84485dd6bb-gr6qv   1/1     Running   0          60s
aws-load-balancer-controller-84485dd6bb-xtwhl   1/1     Running   0          60s
coredns-5b9c98856-5nvz5                         1/1     Running   0          56m
coredns-5b9c98856-k787k                         1/1     Running   0          56m
abhishekveeramalla@aveerama-mac eks % kubectl get deploy -n kube-system
NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
aws-load-balancer-controller   2/2     2            2           67s
coredns                        2/2     2            2           65m
abhishekveeramalla@aveerama-mac eks % kubectl get ingress -n game-2048
NAME           CLASS   HOSTS   ADDRESS                                                                   PORTS   AGE
ingress-2048   alb     *       k8s-game2048-ingress2-e1d9d1f206-1078327551.us-east-1.elb.amazonaws.com   80      7m48s
abhishekveeramalla@aveerama-mac eks %
```

This application load balancer controller should create a ELB.


EC2 > Load balancers

Load balancers (1/1)

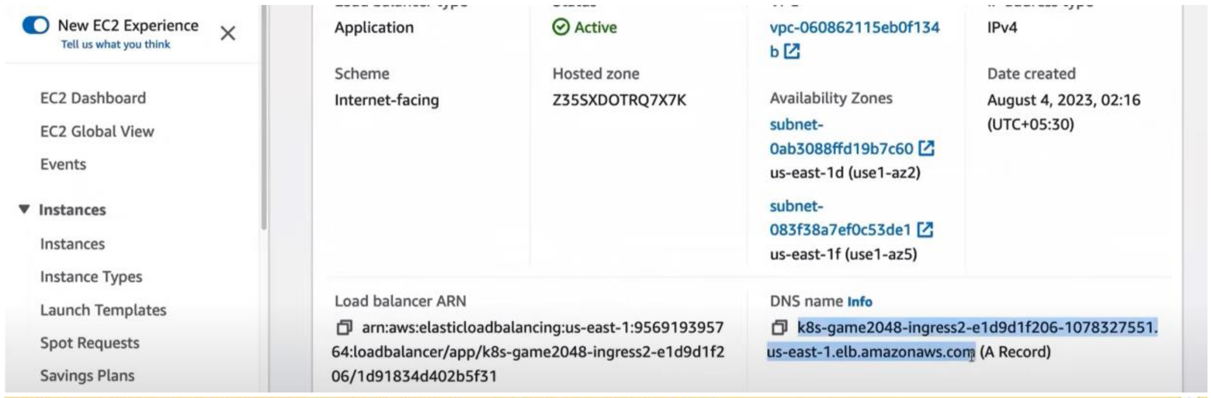
Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

Actions ▼ | Create load balancer ▼

Find resources by attribute or tag

☑	Name	DNS name	State	VPC ID
☑	k8s-game2048-ingress2-e1d9d1f206	k8s-game2048-ingress2-e...	Provisioning	vpc-060862115eb0f13b

Open it and you will find IP.



Application | ✓ Active | vpc-060862115eb0f134b | IPv4

Scheme: Internet-facing

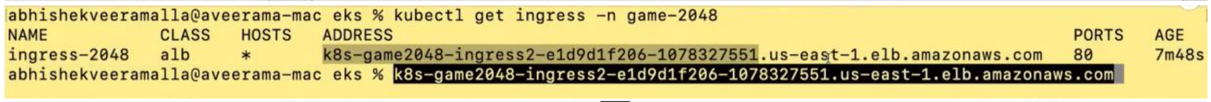
Hosted zone: Z35SXDOTRQ7X7K

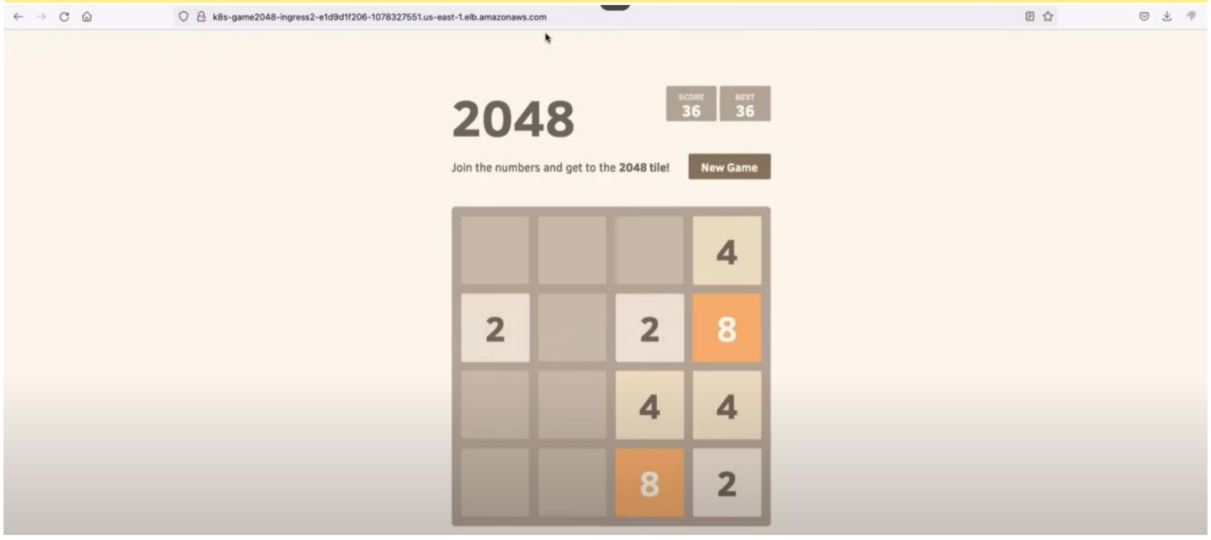
Availability Zones: subnet-0ab3088ffd19b7c60 us-east-1d (use1-az2); subnet-083f38a7ef0c53de1 us-east-1f (use1-az5)

Date created: August 4, 2023, 02:16 (UTC+05:30)

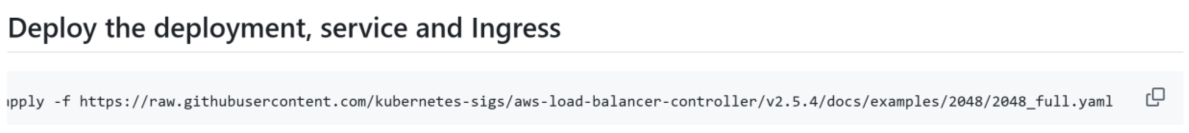
Load balancer ARN: arn:aws:elasticloadbalancing:us-east-1:956919395764:loadbalancer/app/k8s-game2048-ingress2-e1d9d1f206/1d91834d402b5f31

DNS name Info: k8s-game2048-ingress2-e1d9d1f206-1078327551.us-east-1.elb.amazonaws.com (A Record)


```
abhishekveeramalla@averama-mac eks % kubectl get ingress -n game-2048
NAME           CLASS   HOSTS   ADDRESS                                                                     PORTS   AGE
ingress-2048   alb     *       k8s-game2048-ingress2-e1d9d1f206-1078327551.us-east-1.elb.amazonaws.com     80      7m48s
abhishekveeramalla@averama-mac eks % k8s-game2048-ingress2-e1d9d1f206-1078327551.us-east-1.elb.amazonaws.com
```


k8s-game2048-ingress2-e1d9d1f206-1078327551.us-east-1.elb.amazonaws.com
2048
SCORE 36
BEST 36
Join the numbers and get to the 2048 tile!
New Game

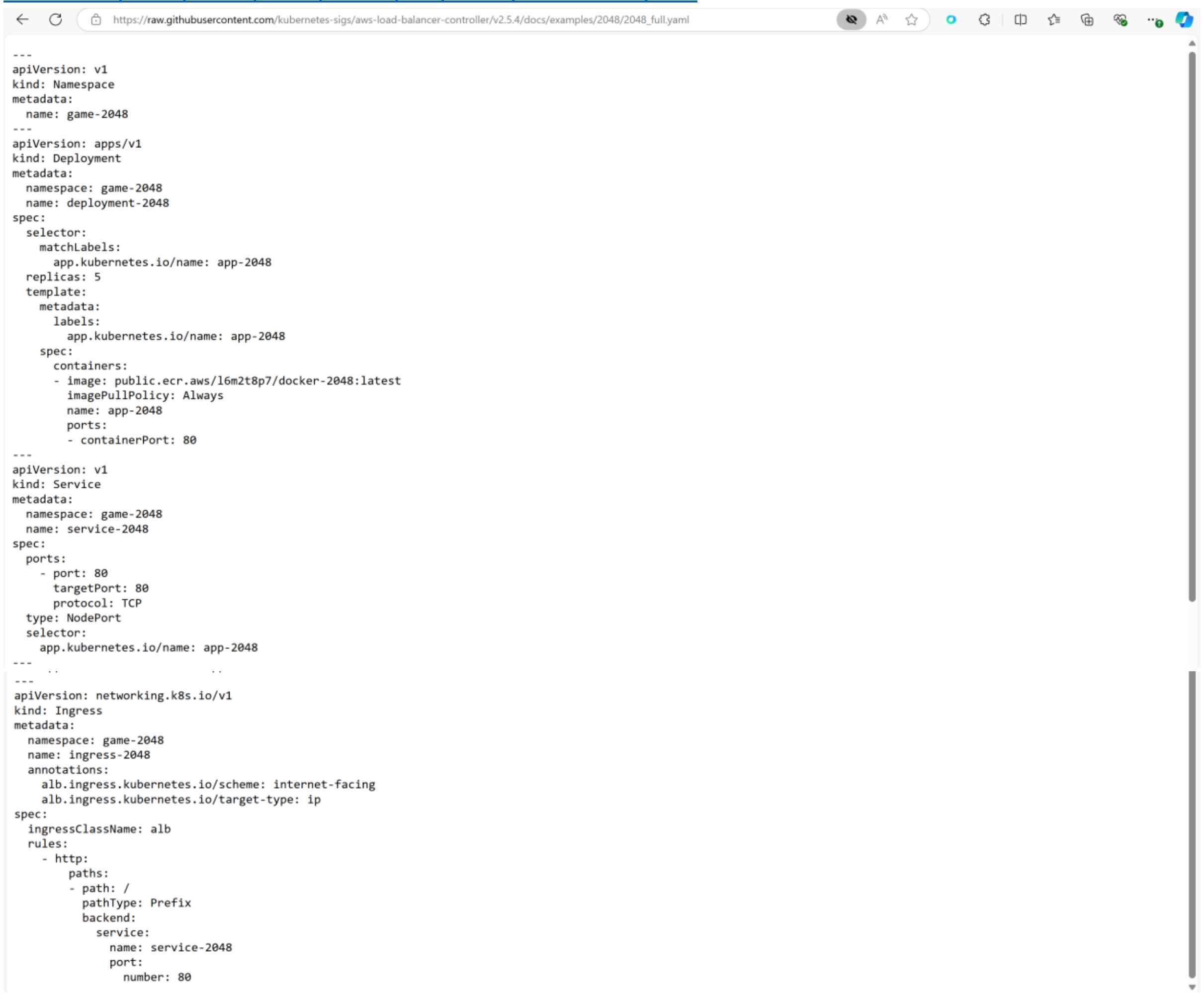
Understanding Deployment, Service & Ingress YAML:



Deploy the deployment, service and Ingress

```
pply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.5.4/docs/examples/2048/2048_full.yaml
```

It is from this Repo: https://github.com/kubernetes-sigs/aws-load-balancer-controller/blob/main/docs/examples/2048/2048_full.yaml



```yaml
---
apiVersion: v1
kind: Namespace
metadata:
  name: game-2048
---
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: game-2048
  name: deployment-2048
spec:
  selector:
    matchLabels:
      app.kubernetes.io/name: app-2048
  replicas: 5
  template:
    metadata:
      labels:
        app.kubernetes.io/name: app-2048
    spec:
      containers:
      - image: public.ecr.aws/l6m2t8p7/docker-2048:latest
        imagePullPolicy: Always
        name: app-2048
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  namespace: game-2048
  name: service-2048
spec:
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
  type: NodePort
  selector:
    app.kubernetes.io/name: app-2048
---
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  namespace: game-2048
  name: ingress-2048
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
spec:
  ingressClassName: alb
  rules:
    - http:
        paths:
        - path: /
          pathType: Prefix
          backend:
            service:
              name: service-2048
              port:
                number: 80
```

Same YAML file in my repo: https://github.com/VarunTej06/EKS-Cluster/blob/main/2048-full.yaml

To be deleted after finishing the demo:

Delete EKS Cluster.

OIDC provider

IAM Policy

IAM Role

Cloudformation stacks

EC2 Load Balancer

Elastic IP

IAM Service Account