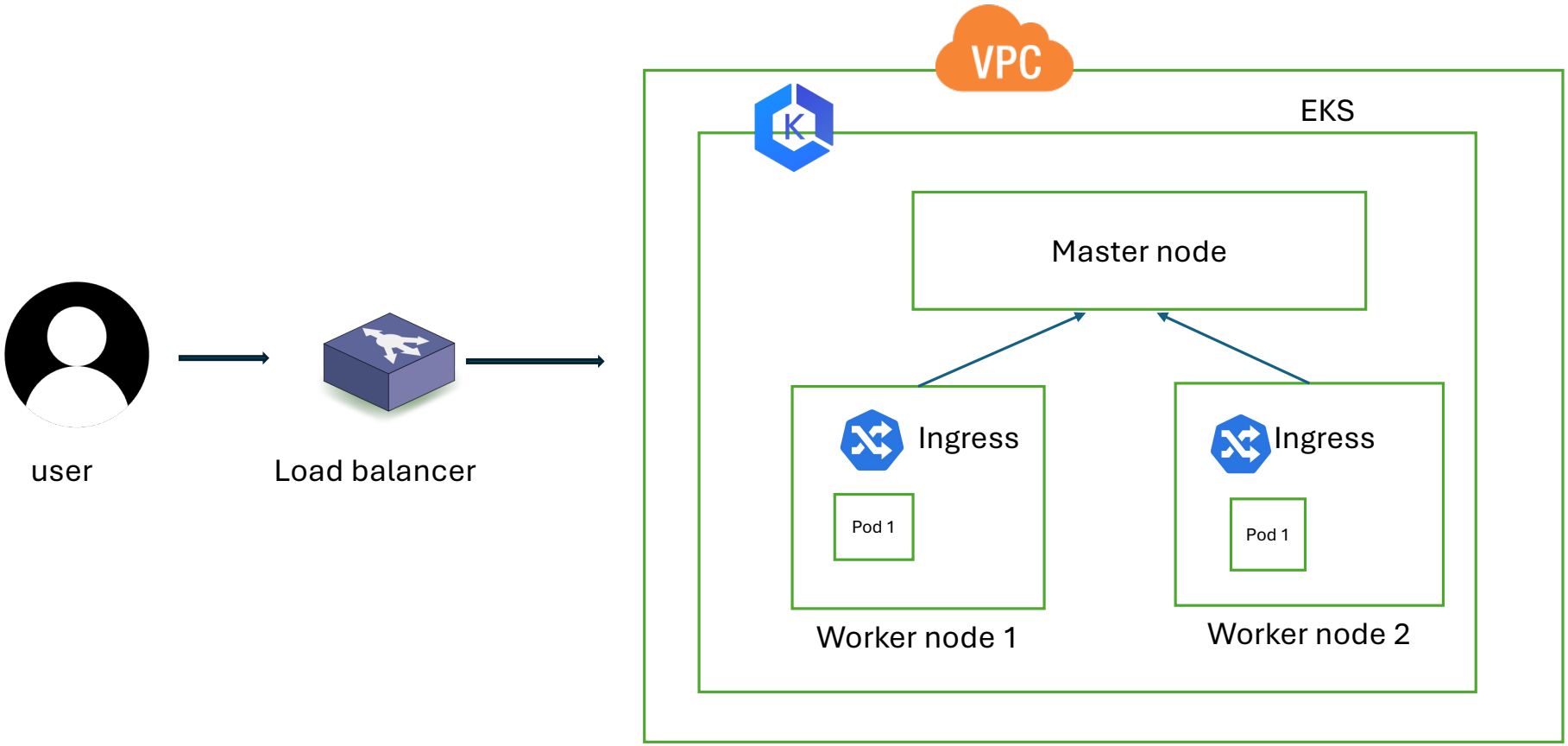
Application Deployment with Ingress on Amazon EKS

Architecture



Services

- ALB
- VPC
- EKS
- Cluster
- Ingress service

Steps and commands

- **Install using Fargate**

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

what is Fargate ?
AWS **Fargate** is a **serverless compute engine** for running Kubernetes **Pods** in EKS **without managing EC2 worker nodes**

Create Fargate profile

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

Update Kubeconfig for EKS Cluster

aws eks update-kubeconfig --region us-east-1 --name demo-cluster

Deploy the deployment, service and Ingress (which is in my local)

Kubectl apply –f 2048_full.yml

Setup alb

Note:

Before setup alb we need to OIDC (OpenID Connect) because the **AWS Load Balancer Controller** uses IAM roles with OIDC to securely authenticate Kubernetes resources with AWS services.

commands to configure IAM OIDC provider

```
export cluster_name=demo-cluster
oidc_id=$(aws eks describe-cluster --name $cluster_name --query "cluster.identity.oidc.issuer" --output text | cut -d '/' -f 5)
```

Check if there is an IAM OIDC provider configured already

aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4\n

If not, run the below command
eksctl utils associate-iam-oidc-provider --cluster $cluster_name --approve

Download IAM policy
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.11.0/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=<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
```

Deploy ALB controller

What is Helm?

- **Helm** is a **package manager for Kubernetes** that helps you define, install, and upgrade applications inside a Kubernetes cluster using **Helm Charts**. It simplifies the deployment of complex applications by managing Kubernetes manifests efficiently.

Add helm repo

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

Update the repo

```
helm repo update eks
```

Install helm

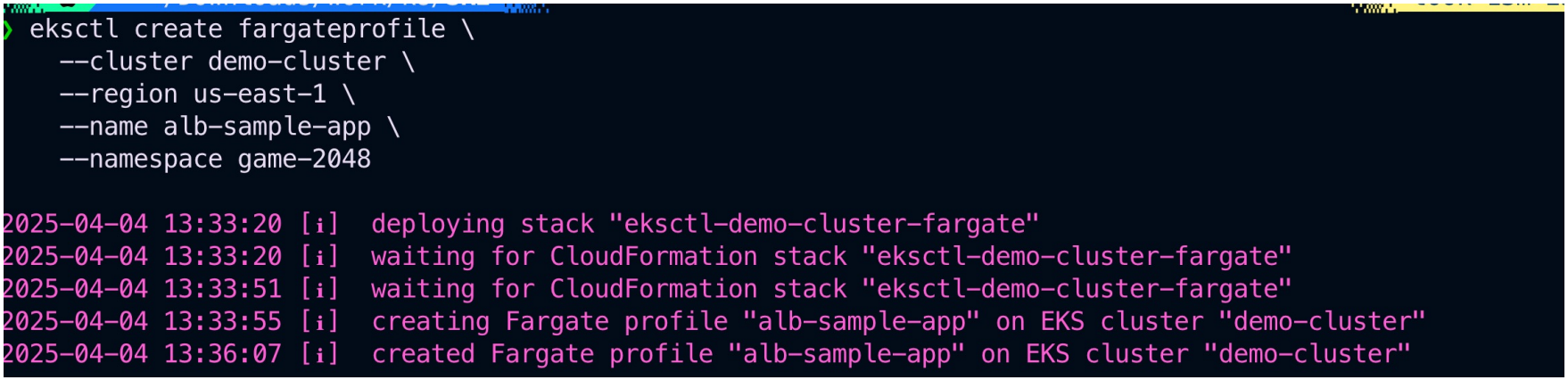
```
helm install aws-load-balancer-controller eks/aws-load-balancer-controller -n kube-system \
--set clusterName=<your-cluster-name> \
--set serviceAccount.create=false \
--set serviceAccount.name=aws-load-balancer-controller \
--set region=<region> \
--set vpcId=<your-vpc-id>
```

Verify that the deployments are running.

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

Output

```
❯ eksctl create cluster --name demo-cluster --region us-east-1
2025-04-04 13:18:44 [ℹ]  eksctl version 0.194.0-dev+02ef28ee3.2024-10-22T20:23:08Z
2025-04-04 13:18:44 [ℹ]  using region us-east-1
2025-04-04 13:18:45 [ℹ]  setting availability zones to [us-east-1c us-east-1d]
2025-04-04 13:18:45 [ℹ]  subnets for us-east-1c - public:192.168.0.0/19 private:192.168.64.0/19
2025-04-04 13:18:45 [ℹ]  subnets for us-east-1d - public:192.168.32.0/19 private:192.168.96.0/19
2025-04-04 13:18:45 [ℹ]  nodegroup "ng-fee5d19f" will use "" [AmazonLinux2/1.30]
2025-04-04 13:18:45 [ℹ]  using Kubernetes version 1.30
2025-04-04 13:18:45 [ℹ]  creating EKS cluster "demo-cluster" in "us-east-1" region with managed nodes
2025-04-04 13:18:45 [ℹ]  will create 2 separate CloudFormation stacks for cluster itself and the initial managed nodegroup
2025-04-04 13:18:45 [ℹ]  if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=u
s-east-1 --cluster=demo-cluster'
2025-04-04 13:18:45 [ℹ]  Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "
demo-cluster" in "us-east-1"
2025-04-04 13:18:45 [ℹ]  CloudWatch logging will not be enabled for cluster "demo-cluster" in "us-east-1"
2025-04-04 13:18:45 [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'
2025-04-04 13:18:45 [ℹ]  default addons vpc-cni, kube-proxy, coredns were not specified, will install them as EKS addons
2025-04-04 13:18:45 [ℹ]
2 sequential tasks: { create cluster control plane "demo-cluster",
    2 sequential sub-tasks: {
        2 sequential sub-tasks: {
            1 task: { create addons },
            wait for control plane to become ready,
        },
        create managed nodegroup "ng-fee5d19f",
    }
}
2025-04-04 13:18:45 [ℹ]  building cluster stack "eksctl-demo-cluster-cluster"
2025-04-04 13:18:46 [ℹ]  deploying stack "eksctl-demo-cluster-cluster"
2025-04-04 13:19:16 [ℹ]  waiting for CloudFormation stack "eksctl-demo-cluster-cluster"
2025-04-04 13:19:47 [ℹ]  waiting for CloudFormation stack "eksctl-demo-cluster-cluster"
2025-04-04 13:20:48 [ℹ]  waiting for CloudFormation stack "eksctl-demo-cluster-cluster"
2025-04-04 13:21:50 [ℹ]  waiting for CloudFormation stack "eksctl-demo-cluster-cluster"
2025-04-04 13:22:51 [ℹ]  waiting for CloudFormation stack "eksctl-demo-cluster-cluster"
```


❯ eksctl create fargateprofile \
--cluster demo-cluster \
--region us-east-1 \
--name alb-sample-app \
--namespace game-2048
2025-04-04 13:33:20 [ℹ] deploying stack "eksctl-demo-cluster-fargate"
2025-04-04 13:33:20 [ℹ] waiting for CloudFormation stack "eksctl-demo-cluster-fargate"
2025-04-04 13:33:51 [ℹ] waiting for CloudFormation stack "eksctl-demo-cluster-fargate"
2025-04-04 13:33:55 [ℹ] creating Fargate profile "alb-sample-app" on EKS cluster "demo-cluster"
2025-04-04 13:36:07 [ℹ] created Fargate profile "alb-sample-app" on EKS cluster "demo-cluster"

Nodes (2) Info

Filter Nodes by property or value

1

Node name	Instance type	Compute	Managed by	Created	Status
ip-192-168-3-155.ec2.internal	m5.large	Node group	ng-fee5d19f	8 minutes ago	Ready
ip-192-168-53-199.ec2.internal	m5.large	Node group	ng-fee5d19f	8 minutes ago	Ready

Node groups (1) Info

Edit | Delete | Add node group

Node groups implement basic compute scaling through EC2 Auto Scaling groups.

Group name	Desired size	AMI release version	Launch template	Status
ng-fee5d19f	2	1.30.9-20250317	eksctl-demo-cluster-nodegroup-ng-fee5d19f (1)	Active

Fargate profiles (1) Info

Edit | Delete | Add Fargate profile

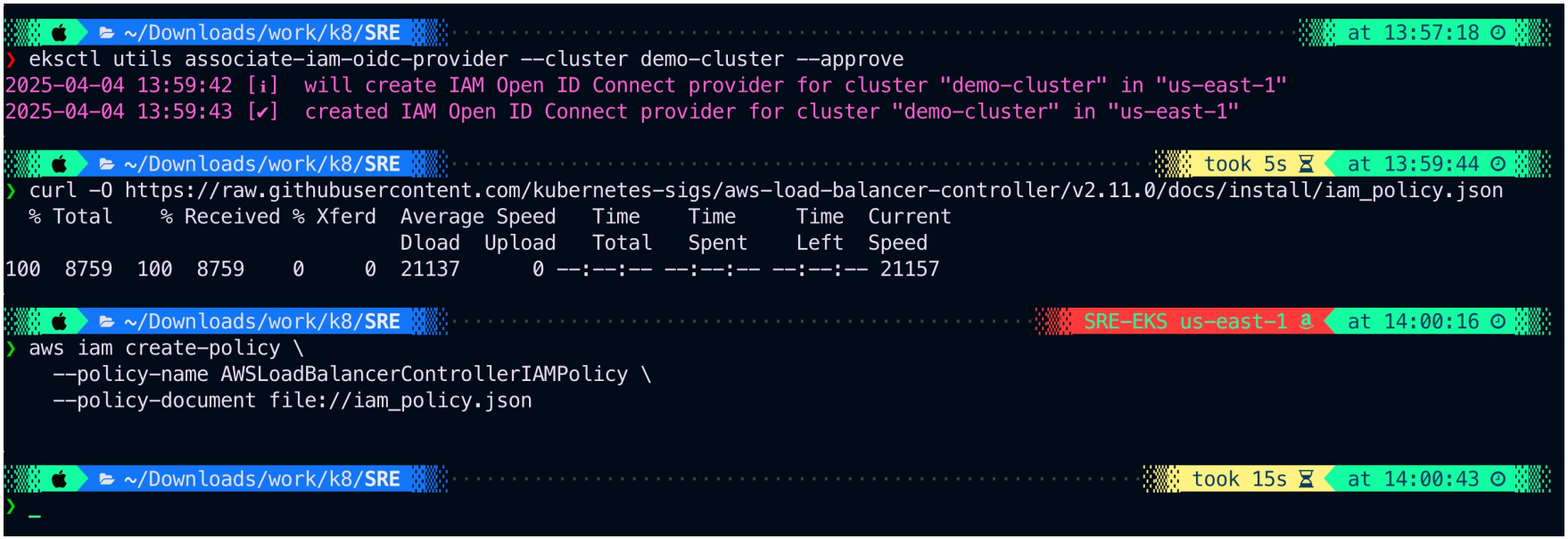
Profile name	Namespaces	Status
alb-sample-app	game-2048	Active

```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
```

```yaml
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
```


~/Downloads/work/k8/SRE
at 15:05:15
ls
2048_full.yaml iam_policy.json
~/Downloads/work/k8/SRE
at demo-cluster
at 15:21:43
kubectl apply -f 2048_full.yaml

Why is OIDC Required for Configuring ALB in AWS EKS?
When setting up **AWS ALB (Application Load Balancer) Controller** in **EKS**, you need **OIDC (OpenID Connect)**. This is required to allow Kubernetes to securely authenticate with AWS services **without using long-term access keys**.

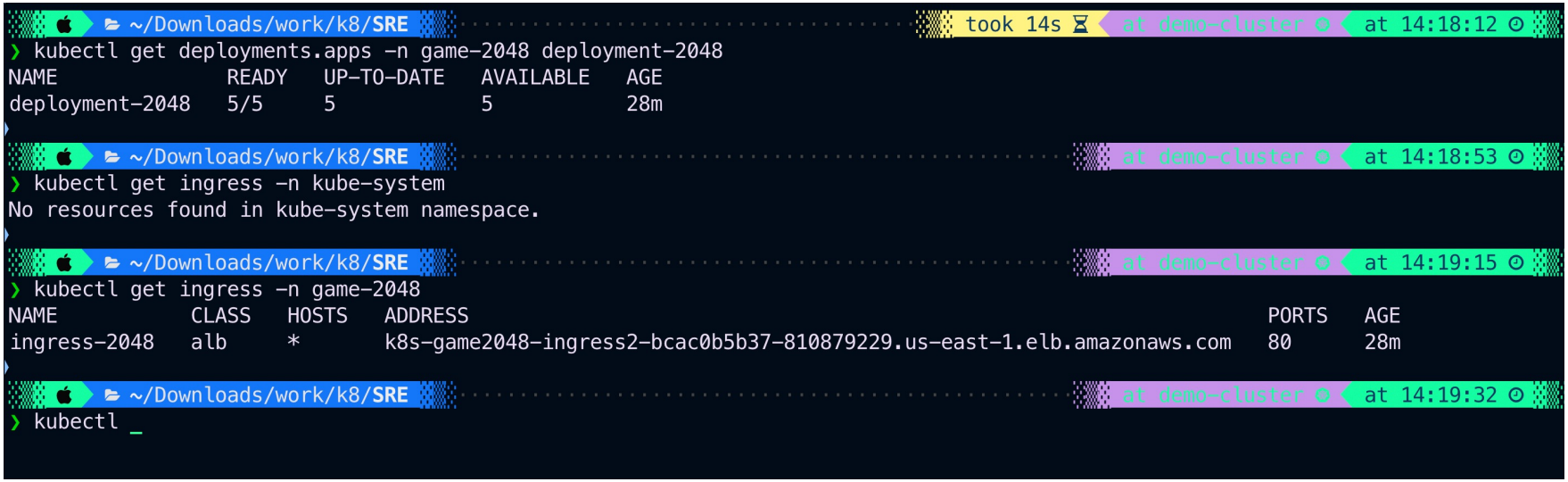


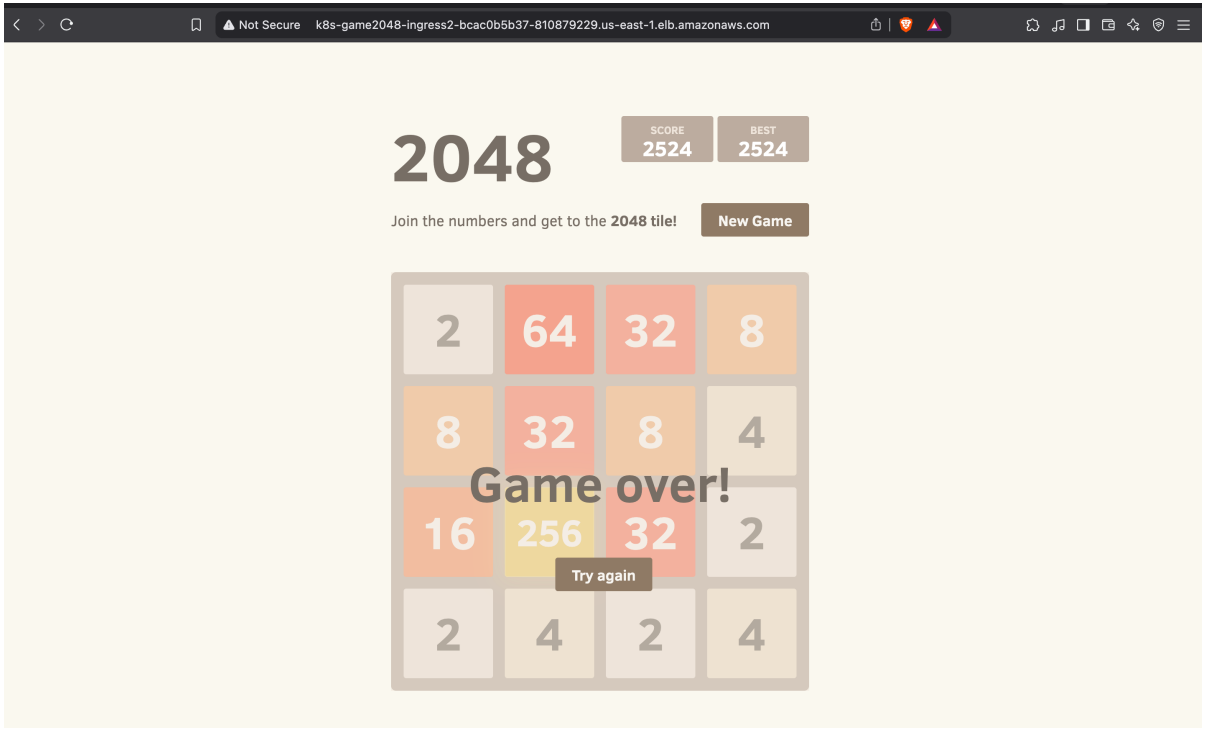
```
 eksctl create iamserviceaccount \
 --cluster=demo-cluster \
 --namespace=kube-system \
 --name=aws-load-balancer-controller \
 --role-name AmazonEKSLoadBalancerControllerRole \
 --attach-policy-arn=arn:aws:iam::235494820580:policy/AWSLoadBalancerControllerIAMPolicy \
 --approve

2025-04-04 14:02:11 [ℹ]  1 iamserviceaccount (kube-system/aws-load-balancer-controller) was included (based on the include/exclude
rules)
2025-04-04 14:02:11 [!]  serviceaccounts that exist in Kubernetes will be excluded, use --override-existing-serviceaccounts to ove
ride
2025-04-04 14:02:11 [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",
   } }2025-04-04 14:02:11 [ℹ]  building iamserviceaccount stack "eksctl-demo-cluster-addon-iamserviceaccount-kube-system-aws-load
balancer-controller"
2025-04-04 14:02:11 [ℹ]  deploying stack "eksctl-demo-cluster-addon-iamserviceaccount-kube-system-aws-load-balancer-controller"
2025-04-04 14:02:12 [ℹ]  waiting for CloudFormation stack "eksctl-demo-cluster-addon-iamserviceaccount-kube-system-aws-load-balanc
r-controller"
2025-04-04 14:02:42 [ℹ]  waiting for CloudFormation stack "eksctl-demo-cluster-addon-iamserviceaccount-kube-system-aws-load-balanc
r-controller"
2025-04-04 14:02:43 [ℹ]  created serviceaccount "kube-system/aws-load-balancer-controller"
```

```
~/Downloads/work/k8/SRE                                                    at demo-cl
❯ 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-08e4325e6891e1b8f

NAME: aws-load-balancer-controller
LAST DEPLOYED: Fri Apr  4 14:18:02 2025
NAMESPACE: kube-system
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
AWS Load Balancer controller installed!
```


~/Downloads/work/k8/SRE
took 14s
at demo-cluster
at 14:18:12
❯ kubectl get deployments.apps -n game-2048 deployment-2048
NAME READY UP-TO-DATE AVAILABLE AGE
deployment-2048 5/5 5 5 28m
~/Downloads/work/k8/SRE
at demo-cluster
at 14:18:53
❯ kubectl get ingress -n kube-system
No resources found in kube-system namespace.
~/Downloads/work/k8/SRE
at demo-cluster
at 14:19:15
❯ kubectl get ingress -n game-2048
NAME CLASS HOSTS ADDRESS PORTS AGE
ingress-2048 alb * k8s-game2048-ingress2-bcac0b5b37-810879229.us-east-1.elb.amazonaws.com 80 28m
~/Downloads/work/k8/SRE
at demo-cluster
at 14:19:32
❯ kubectl _


Not Secure
k8s-game2048-ingress2-bcac0b5b37-810879229.us-east-1.elb.amazonaws.com
2048
SCORE
2524
BEST
2524
Join the numbers and get to the 2048 tile!
New Game
2 64 32 8
8 32 8 4
16 256 32 2
2 4 2 4
Game over!
Try again