Deploy 2048 Game on Amazon EKS

> **Project:** End-to-end deployment of the `2048` sample game onto an Amazon EKS cluster.
>
> **Goal:** Show how to create an EKS cluster, install the AWS Load Balancer Controller (ALB), deploy the 2048 app, and expose it to the internet using an Ingress backed by an AWS Application Load Balancer (ALB).

🚀 Overview

This repo documents a repeatable, production-minded workflow to deploy the open-source `2048` web game to EKS. It uses:

- `eksctl` to create the cluster and bootstrap OIDC (IRSA)
- `helm` to install the AWS Load Balancer Controller (ALB Ingress)
- Kubernetes manifests (Deployment, Service, Ingress) to run and expose the app

The instructions are copy-paste ready for an Ubuntu or macOS terminal with AWS credentials configured.

🛖 What you will build

```
User
  |
  └──> AWS ALB (internet-facing)
         |
         └──> EKS (aws-load-balancer-controller)
               └──> Namespace: game-2048
                     ├─ Deployment (2048 app, 3 replicas)
                     ├─ Service (ClusterIP)
                     └─ Ingress (ALB) -> routes to Service
```

Prerequisites

- An AWS account with permission to create EKS, IAM, EC2, ELB, and related resources.
- `aws` CLI v2 installed and configured (`aws configure`).
- `eksctl` installed.
- `kubectl` installed.
- `helm` v3 installed.
- A supported region (example uses `ap-south-1`, change `REGION` as needed).

> **Tip:** Use a non-production account or clean up when finished to avoid costs.

Step-by-step

All commands should be run in a terminal. Replace variables as needed.

1) Environment variables

```
export REGION=ap-south-1        # change to your preferred region
export CLUSTER=eks-2048-demo
```

2) Create EKS cluster (managed nodes)

```
eksctl create cluster \
  --name $CLUSTER \
  --region $REGION \
  --version 1.29 \
  --nodegroup-name ng1 \
  --node-type t3.medium \
  --nodes 2 \
  --nodes-min 2 \
  --nodes-max 4 \
  --managed

# update kubeconfig (eksctl normally does this for you)
aws eks update-kubeconfig --region $REGION --name $CLUSTER
kubectl get nodes
```

The cluster creation can take 10–20 minutes.

3) Associate IAM OIDC provider (IRSA)

```
eksctl utils associate-iam-oidc-provider --cluster $CLUSTER --region $REGION
--approve
```

4) Create IAM policy for AWS Load Balancer Controller

```
# Download the official policy JSON (version used in this guide)
curl -o iam_policy.json \
  https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-
controller/v2.7.1/docs/install/iam_policy.json

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

POLICY_ARN=$(aws iam list-policies --query "Policies[?
PolicyName=='AWSLoadBalancerControllerIAMPolicy'].Arn | [0]" --output text)
```

5) Create IRSA ServiceAccount for the controller

```
eksctl create iamserviceaccount \
  --cluster $CLUSTER \
  --region $REGION \
  --namespace kube-system \
  --name aws-load-balancer-controller \
  --attach-policy-arn "$POLICY_ARN" \
  --approve
```

6) Install AWS Load Balancer Controller via Helm

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

helm upgrade --install aws-load-balancer-controller eks/aws-load-balancer-
controller \
  -n kube-system \
  --set clusterName=$CLUSTER \
  --set serviceAccount.create=false \
  --set serviceAccount.name=aws-load-balancer-controller

# verify
kubectl get deployment -n kube-system aws-load-balancer-controller
kubectl get pods -n kube-system | grep aws-load-balancer-controller
```

7) Deploy the 2048 app manifests

Create a file named `2048-manifests.yaml` with the following content and apply it.

```
# 2048-manifests.yaml
---
apiVersion: v1
kind: Namespace
metadata:
  name: game-2048
---
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: game-2048
  name: deployment-2048
spec:
  replicas: 3
  selector:
    matchLabels:
      app.kubernetes.io/name: app-2048
  template:
```

```yaml
    metadata:
      labels:
        app.kubernetes.io/name: app-2048
    spec:
      containers:
        - name: app-2048
          image: public.ecr.aws/l6m2t8p7/docker-2048:latest
          ports:
            - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  namespace: game-2048
  name: service-2048
spec:
  type: ClusterIP
  selector:
    app.kubernetes.io/name: app-2048
  ports:
    - port: 80
      targetPort: 80
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  namespace: game-2048
  name: ingress-2048
  annotations:
    kubernetes.io/ingress.class: alb
    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
```

Apply the manifests:

```
kubectl apply -f 2048-manifests.yaml
kubectl get all -n game-2048
kubectl get ingress -n game-2048
```

The Ingress will create an ALB. It can take a few minutes for the ALB DNS name to appear in the `ADDRESS` field of the Ingress.

8) Access the app

```
kubectl get ingress -n game-2048 -o wide
# Look for the ALB DNS (ADDRESS). Open it in your browser.
```

If you used `host` rules in the Ingress, point your domain's DNS (A or CNAME) to the ALB DNS.

Verification checklist

- `kubectl get pods -n kube-system` → `aws-load-balancer-controller` running
- `kubectl get pods -n game-2048` → 3 replicas Ready
- `kubectl get svc -n game-2048` → `service-2048` exists (ClusterIP)
- `kubectl get ingress -n game-2048` → ADDRESS shows ALB DNS
- Hit the ALB DNS in a browser and confirm the 2048 game loads

---

Troubleshooting

ALB never appears / Ingress stuck

- Check the controller logs:

```
kubectl logs -n kube-system deployment/aws-load-balancer-controller
```

- Ensure the ServiceAccount is bound to the correct IAM policy (IRSA).
- Confirm the Ingress has the annotation `kubernetes.io/ingress.class: alb` or `ingressClassName: alb`.

Pods have ImagePullBackOff

- Ensure nodes are compatible (x86 vs ARM) with the container image architecture.

DNS/TLS issues

- If using TLS or cert-manager, ensure ALB is reachable publicly for HTTP-01 challenges or use ACM + annotations for TLS.

🚞Cleanup (remove resources & avoid charges)

```
kubectl delete -f 2048-manifests.yaml
helm uninstall aws-load-balancer-controller -n kube-system
eksctl delete cluster --name $CLUSTER --region $REGION
# Optionally remove the IAM policy if you created it manually
aws iam delete-policy --policy-arn $POLICY_ARN
```

🏠Cost & Notes

- Running an EKS cluster, EC2 worker nodes, ALB, and NAT/RDS (if used) will incur charges. Use small instance types and delete the cluster when done.
- Prefer testing in a sandbox account.

References

- AWS EKS and AWS Load Balancer Controller official docs (see upstream for latest versions and policy JSON).

Author

Anil Sai Telagarapu

License