EKS ENVIRONMENT

- **Prerequisites**
  - Linux distribution (CentOS 9 in this case)
  - AWS CLI installed
  - EKSCTL installed
  - helm installed
  - Root account in AWS platform

- **Setting up the AWS EKS environment**
  - Setting up the connection with AWS console executing the next command:

aws configure

```
[luis@vbox ~]$ aws configure
AWS Access Key ID [****************FN4L]: [redacted]
AWS Secret Access Key [****************Bxtg]: [redacted]
Default region name [us-east-1]:
Default output format [json]:
```

  - Creating a EKS cluster with the next command:

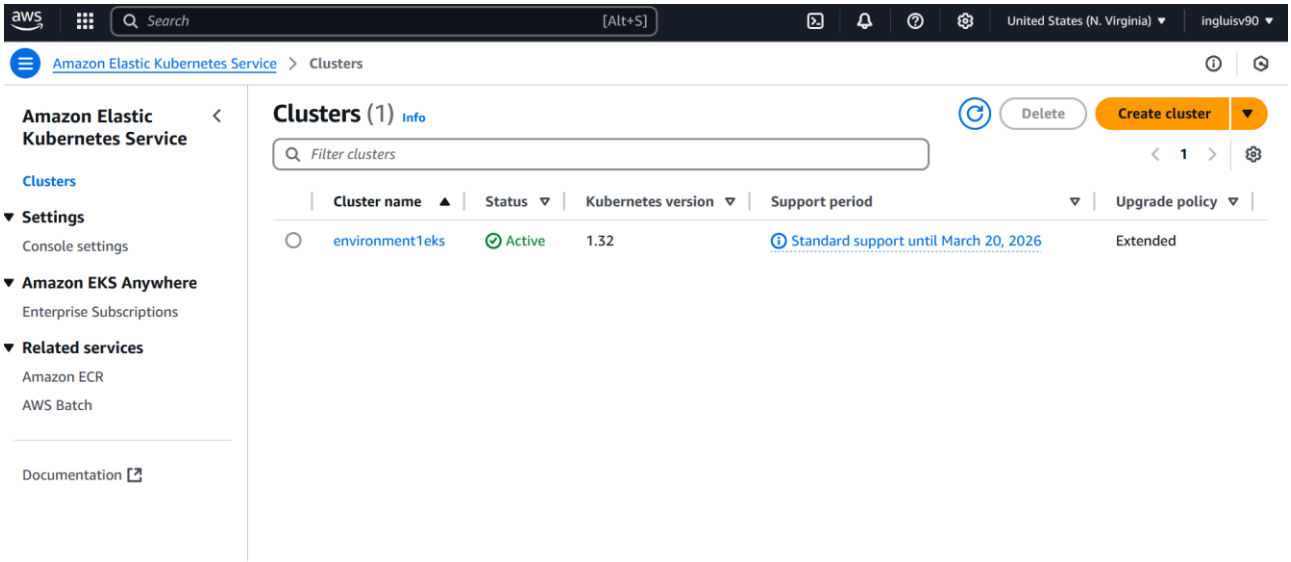
eksctl create cluster --name environment1eks --region us-east-1 --fargate

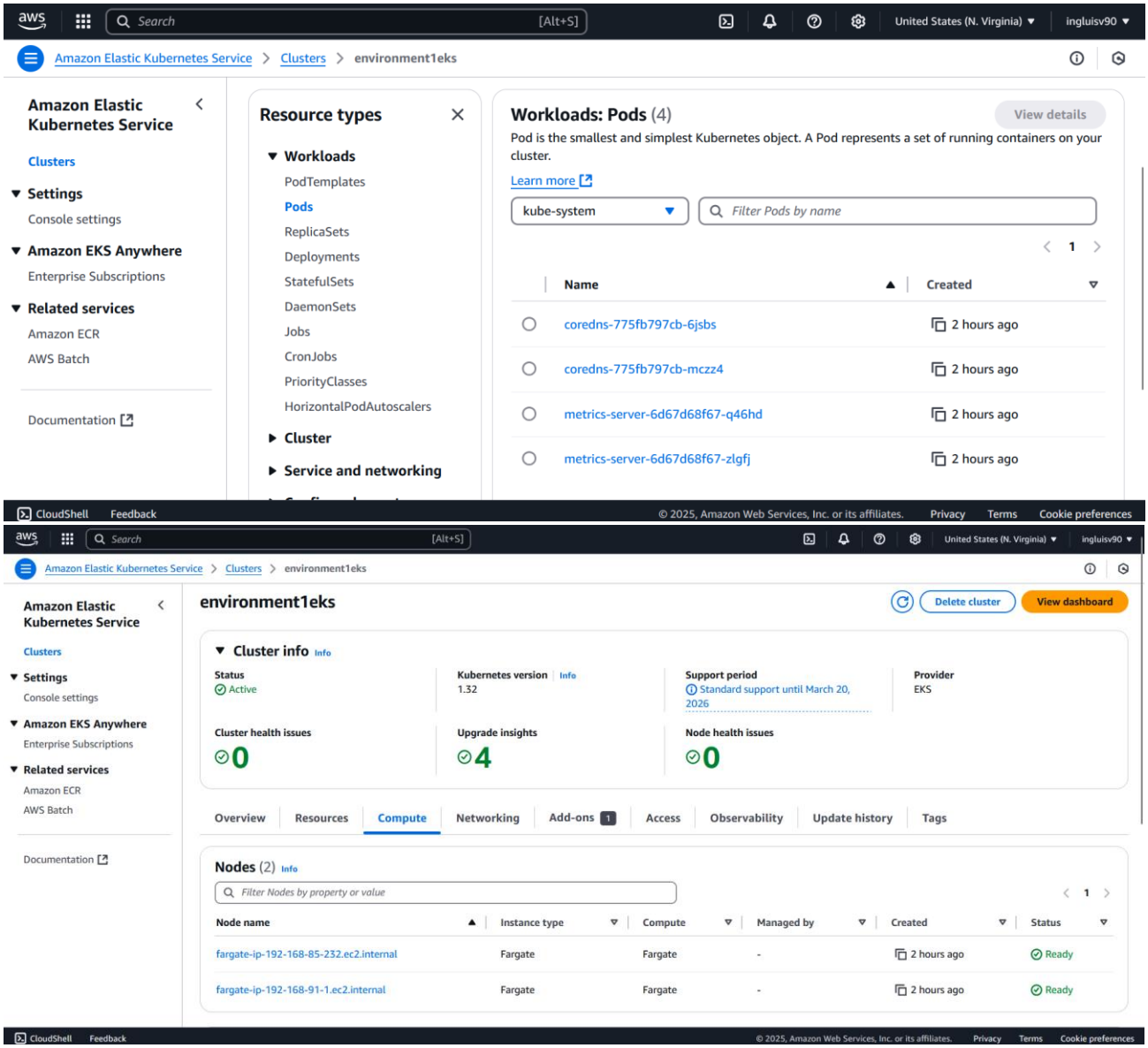
```
[luis@vbox ~]$ eksctl create cluster --name environment1eks --region us-east-1 --fargate
2025-04-24 13:40:36 [ℹ]  eksctl version 0.207.0
2025-04-24 13:40:36 [ℹ]  using region us-east-1
2025-04-24 13:40:37 [ℹ]  setting availability zones to [us-east-1f us-east-1d]
2025-04-24 13:40:37 [ℹ]  subnets for us-east-1f - public:192.168.0.0/19 private:192.168.64.0/19
2025-04-24 13:40:37 [ℹ]  subnets for us-east-1d - public:192.168.32.0/19 private:192.168.96.0/19
2025-04-24 13:40:37 [ℹ]  using Kubernetes version 1.32
2025-04-24 13:40:37 [ℹ]  creating EKS cluster "environment1eks" in "us-east-1" region with Fargate profile
2025-04-24 13:40:37 [ℹ]  if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --regi
on=us-east-1 --cluster=environment1eks'
2025-04-24 13:40:37 [ℹ]  Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for clust
er "environment1eks" in "us-east-1"
2025-04-24 13:40:37 [ℹ]  CloudWatch logging will not be enabled for cluster "environment1eks" in "us-east-1"
2025-04-24 13:40:37 [ℹ]  you can enable it with 'eksctl utils update-cluster-logging --enable-types={SPECIFY-YOUR-LOG-TYPES-HE
RE (e.g. all)} --region=us-east-1 --cluster=environment1eks'
2025-04-24 13:40:37 [ℹ]  default addons kube-proxy, coredns, metrics-server, vpc-cni were not specified, will install them as
EKS addons
2025-04-24 13:40:37 [ℹ]
2 sequential tasks: { create cluster control plane "environment1eks",
    3 sequential sub-tasks: {
        1 task: { create addons },
        wait for control plane to become ready,
        create fargate profiles,
    }
}
2025-04-24 13:40:37 [ℹ]  building cluster stack "eksctl-environment1eks-cluster"
2025-04-24 13:40:38 [ℹ]  deploying stack "eksctl-environment1eks-cluster"
2025-04-24 13:41:08 [ℹ]  waiting for CloudFormation stack "eksctl-environment1eks-cluster"
2025-04-24 13:41:38 [ℹ]  waiting for CloudFormation stack "eksctl-environment1eks-cluster"
2025-04-24 13:42:39 [ℹ]  waiting for CloudFormation stack "eksctl-environment1eks-cluster"
2025-04-24 13:43:40 [ℹ]  waiting for CloudFormation stack "eksctl-environment1eks-cluster"
2025-04-24 13:44:40 [ℹ]  waiting for CloudFormation stack "eksctl-environment1eks-cluster"
2025-04-24 13:45:41 [ℹ]  waiting for CloudFormation stack "eksctl-environment1eks-cluster"
```

```
2025-04-24 13:46:41 [i]  waiting for CloudFormation stack "eksctl-environment1eks-cluster"
2025-04-24 13:47:42 [i]  waiting for CloudFormation stack "eksctl-environment1eks-cluster"
2025-04-24 13:48:42 [i]  waiting for CloudFormation stack "eksctl-environment1eks-cluster"
2025-04-24 13:48:44 [i]  creating addon: kube-proxy
2025-04-24 13:48:45 [i]  successfully created addon: kube-proxy
2025-04-24 13:48:45 [i]  creating addon: coredns
2025-04-24 13:48:46 [i]  successfully created addon: coredns
2025-04-24 13:48:46 [i]  creating addon: metrics-server
2025-04-24 13:48:47 [i]  successfully created addon: metrics-server
2025-04-24 13:48:47 [!]  recommended policies were found for "vpc-cni" addon, but since OIDC is disabled on the cluster, eksct
l cannot configure the requested permissions; the recommended way to provide IAM permissions for "vpc-cni" addon is via pod id
entity associations; after addon creation is completed, add all recommended policies to the config file, under `addon.PodIdent
ityAssociations`, and run `eksctl update addon`
2025-04-24 13:48:47 [i]  creating addon: vpc-cni
2025-04-24 13:48:48 [i]  successfully created addon: vpc-cni
2025-04-24 13:50:50 [i]  creating Fargate profile "fp-default" on EKS cluster "environment1eks"
2025-04-24 13:53:01 [i]  created Fargate profile "fp-default" on EKS cluster "environment1eks"
2025-04-24 13:53:31 [i]  "coredns" is now schedulable onto Fargate
2025-04-24 13:54:36 [i]  "coredns" is now scheduled onto Fargate
2025-04-24 13:54:36 [i]  "coredns" pods are now scheduled onto Fargate
2025-04-24 13:54:36 [i]  waiting for the control plane to become ready
2025-04-24 13:54:38 [✔]  saved kubeconfig as "/home/luis/.kube/config"
2025-04-24 13:54:38 [i]  no tasks
2025-04-24 13:54:38 [✔]  all EKS cluster resources for "environment1eks" have been created
2025-04-24 13:54:42 [i]  kubectl command should work with "/home/luis/.kube/config", try 'kubectl get nodes'
2025-04-24 13:54:42 [✔]  EKS cluster "environment1eks" in "us-east-1" region is ready
```

This command will trigger the creation of service roles, public subnet, and private subnet. In the private subnet we will place the application.

- Checking the AWS console we can see the EKS cluster created





- Downloading de kubeconfig file with the next command:

aws eks update-kubeconfig --name environment1eks --region us-east-1

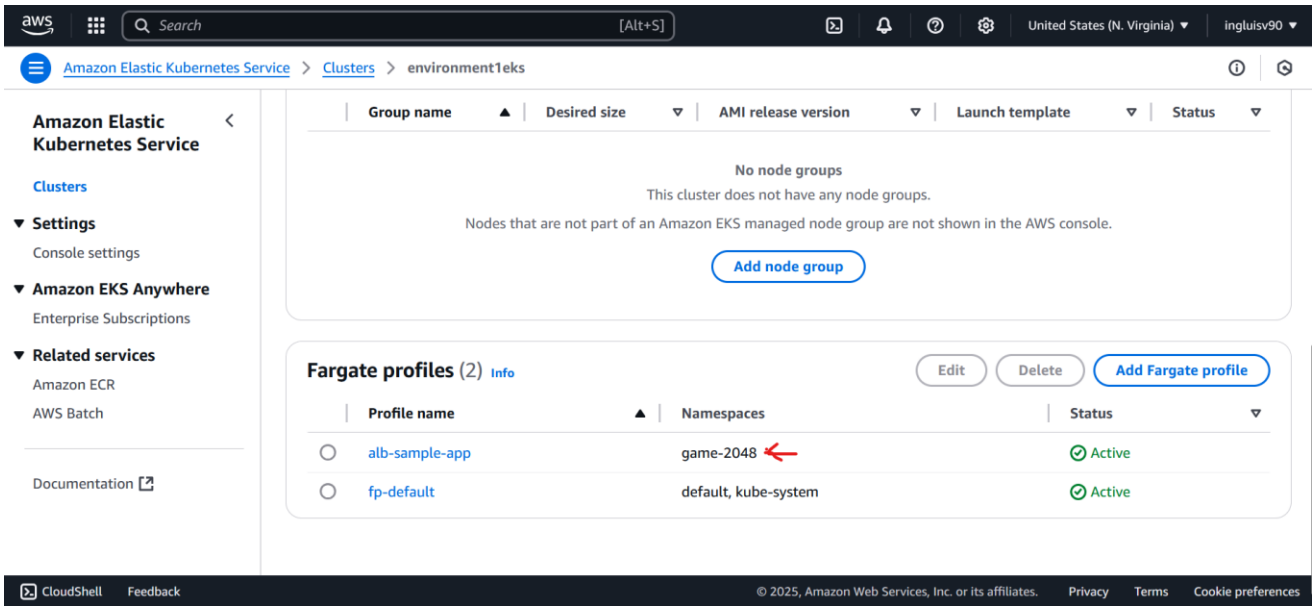
```
[luis@vbox ~]$ aws eks update-kubeconfig --name environment1eks --region us-east-1
Added new context arn:aws:eks:us-east-1:390402579510:cluster/environment1eks to /home/luis/.kube/config
[luis@vbox ~]$
```

- Creating a fargate profile with the next command:

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

```
[luis@vbox ~]$ eksctl create fargateprofile \
    --cluster environment1eks \
    --region us-east-1 \
    --name alb-sample-app \
    --namespace game-2048
2025-04-24 16:17:51 [ℹ]  creating Fargate profile "alb-sample-app" on EKS cluster "environment1eks"
2025-04-24 16:18:25 [ℹ]  created Fargate profile "alb-sample-app" on EKS cluster "environment1eks"
[luis@vbox ~]$
```

Checking in the AWS console



- Deploying the application from repository with the next command:

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

Here is the content of the deployment yaml file (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
```

This is the output

```
[luis@vbox ~]$ kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.5.4/docs/exa
mples/2048/2048_full.yaml
namespace/game-2048 created
deployment.apps/deployment-2048 created
service/service-2048 created
ingress.networking.k8s.io/ingress-2048 created
[luis@vbox ~]$
```

Checking the pods status with:

kubectl get pods -n game-2048

```
[luis@vbox ~]$ kubectl get pods -n game-2048
NAME                                READY   STATUS    RESTARTS   AGE
deployment-2048-bdbddc878-27xhf   1/1     Running   0          67s
deployment-2048-bdbddc878-2b77h   1/1     Running   0          67s
deployment-2048-bdbddc878-cw8wm   1/1     Running   0          67s
deployment-2048-bdbddc878-h7jbr   1/1     Running   0          67s
deployment-2048-bdbddc878-tzxx9   1/1     Running   0          67s
```

Checking the service with:
kubectl get svc -n game-2048

```
[luis@vbox ~]$ kubectl get svc -n game-2048
NAME           TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
service-2048   NodePort   10.100.154.41   <none>        80:32048/TCP   2m25s
```

Checking the ingress created with:
kubectl get ingress -n game-2048

```
[luis@vbox ~]$ kubectl get ingress -n game-2048
NAME           CLASS   HOSTS   ADDRESS   PORTS   AGE
ingress-2048   alb     *                 80      4m33s
```

- Creating an ingress controller to get the proper access, with the next command:

eksctl utils associate-iam-oidc-provider --cluster environment1eks --approve

```
[luis@vbox ~]$ eksctl utils associate-iam-oidc-provider --cluster environment1eks --approve
2025-04-24 16:43:45 [ℹ]  will create IAM Open ID Connect provider for cluster "environment1eks" in "us-east-1"
2025-04-24 16:43:46 [✔]  created IAM Open ID Connect provider for cluster "environment1eks" in "us-east-1"
[luis@vbox ~]$
```

- Downloading IAM policy

curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.11.0/docs/install/iam_policy.json

iam_policy.json content here:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "elasticloadbalancing.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeAddresses",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeInternetGateways",
                "ec2:DescribeVpcs",
                "ec2:DescribeVpcPeeringConnections",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeInstances",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeTags",
                "ec2:GetCoipPoolUsage",
                "ec2:DescribeCoipPools",
                "ec2:GetSecurityGroupsForVpc",
                "elasticloadbalancing:DescribeLoadBalancers",
                "elasticloadbalancing:DescribeLoadBalancerAttributes",
                "elasticloadbalancing:DescribeListeners",
                "elasticloadbalancing:DescribeListenerCertificates",
                "elasticloadbalancing:DescribeSSLPolicies",
                "elasticloadbalancing:DescribeRules",
                "elasticloadbalancing:DescribeTargetGroups",
                "elasticloadbalancing:DescribeTargetGroupAttributes",
                "elasticloadbalancing:DescribeTargetHealth",
                "elasticloadbalancing:DescribeTags",
                "elasticloadbalancing:DescribeTrustStores",
                "elasticloadbalancing:DescribeListenerAttributes",
                "elasticloadbalancing:DescribeCapacityReservation"
            ],
```

```
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cognito-idp:DescribeUserPoolClient",
                "acm:ListCertificates",
                "acm:DescribeCertificate",
                "iam:ListServerCertificates",
                "iam:GetServerCertificate",
                "waf-regional:GetWebACL",
                "waf-regional:GetWebACLForResource",
                "waf-regional:AssociateWebACL",
                "waf-regional:DisassociateWebACL",
                "wafv2:GetWebACL",
                "wafv2:GetWebACLForResource",
                "wafv2:AssociateWebACL",
                "wafv2:DisassociateWebACL",
                "shield:GetSubscriptionState",
                "shield:DescribeProtection",
                "shield:CreateProtection",
                "shield:DeleteProtection"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:RevokeSecurityGroupIngress"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateSecurityGroup"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": "arn:aws:ec2:*:*:security-group/*",
            "Condition": {
                "StringEquals": {
```

```
            "ec2:CreateAction": "CreateSecurityGroup"
        },
        "Null": {
            "aws:RequestTag/elbv2.k8s.aws/cluster": "false"
        }
    }
},
{
    "Effect": "Allow",
    "Action": [
        "ec2:CreateTags",
        "ec2:DeleteTags"
    ],
    "Resource": "arn:aws:ec2:*:*:security-group/*",
    "Condition": {
        "Null": {
            "aws:RequestTag/elbv2.k8s.aws/cluster": "true",
            "aws:ResourceTag/elbv2.k8s.aws/cluster": "false"
        }
    }
},
{
    "Effect": "Allow",
    "Action": [
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:RevokeSecurityGroupIngress",
        "ec2:DeleteSecurityGroup"
    ],
    "Resource": "*",
    "Condition": {
        "Null": {
            "aws:ResourceTag/elbv2.k8s.aws/cluster": "false"
        }
    }
},
{
    "Effect": "Allow",
    "Action": [
        "elasticloadbalancing:CreateLoadBalancer",
        "elasticloadbalancing:CreateTargetGroup"
    ],
    "Resource": "*",
    "Condition": {
        "Null": {
            "aws:RequestTag/elbv2.k8s.aws/cluster": "false"
        }
    }
}
```

```
},
{
    "Effect": "Allow",
    "Action": [
        "elasticloadbalancing:CreateListener",
        "elasticloadbalancing:DeleteListener",
        "elasticloadbalancing:CreateRule",
        "elasticloadbalancing:DeleteRule"
    ],
    "Resource": "*"
},
{
    "Effect": "Allow",
    "Action": [
        "elasticloadbalancing:AddTags",
        "elasticloadbalancing:RemoveTags"
    ],
    "Resource": [
        "arn:aws:elasticloadbalancing:*:*:targetgroup/*/*",
        "arn:aws:elasticloadbalancing:*:*:loadbalancer/net/*/*",
        "arn:aws:elasticloadbalancing:*:*:loadbalancer/app/*/*"
    ],
    "Condition": {
        "Null": {
            "aws:RequestTag/elbv2.k8s.aws/cluster": "true",
            "aws:ResourceTag/elbv2.k8s.aws/cluster": "false"
        }
    }
},
{
    "Effect": "Allow",
    "Action": [
        "elasticloadbalancing:AddTags",
        "elasticloadbalancing:RemoveTags"
    ],
    "Resource": [
        "arn:aws:elasticloadbalancing:*:*:listener/net/*/*/*",
        "arn:aws:elasticloadbalancing:*:*:listener/app/*/*/*",
        "arn:aws:elasticloadbalancing:*:*:listener-rule/net/*/*/*",
        "arn:aws:elasticloadbalancing:*:*:listener-rule/app/*/*/*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "elasticloadbalancing:ModifyLoadBalancerAttributes",
        "elasticloadbalancing:SetIpAddressType",
        "elasticloadbalancing:SetSecurityGroups",
```

```
            "elasticloadbalancing:SetSubnets",
            "elasticloadbalancing:DeleteLoadBalancer",
            "elasticloadbalancing:ModifyTargetGroup",
            "elasticloadbalancing:ModifyTargetGroupAttributes",
            "elasticloadbalancing:DeleteTargetGroup",
            "elasticloadbalancing:ModifyListenerAttributes",
            "elasticloadbalancing:ModifyCapacityReservation"
        ],
        "Resource": "*",
        "Condition": {
            "Null": {
                "aws:ResourceTag/elbv2.k8s.aws/cluster": "false"
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": [
            "elasticloadbalancing:AddTags"
        ],
        "Resource": [
            "arn:aws:elasticloadbalancing:*:*:targetgroup/*/*",
            "arn:aws:elasticloadbalancing:*:*:loadbalancer/net/*/*",
            "arn:aws:elasticloadbalancing:*:*:loadbalancer/app/*/*"
        ],
        "Condition": {
            "StringEquals": {
                "elasticloadbalancing:CreateAction": [
                    "CreateTargetGroup",
                    "CreateLoadBalancer"
                ]
            },
            "Null": {
                "aws:RequestTag/elbv2.k8s.aws/cluster": "false"
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": [
            "elasticloadbalancing:RegisterTargets",
            "elasticloadbalancing:DeregisterTargets"
        ],
        "Resource": "arn:aws:elasticloadbalancing:*:*:targetgroup/*/*"
    },
    {
        "Effect": "Allow",
        "Action": [
                "elasticloadbalancing:SetWebAcl",
                "elasticloadbalancing:ModifyListener",
                "elasticloadbalancing:AddListenerCertificates",
                "elasticloadbalancing:RemoveListenerCertificates",
                "elasticloadbalancing:ModifyRule"
            ],
            "Resource": "*"
        }
    ]
}
```

Executing the command:

```
[luis@vbox ~]$ curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.11.0/docs/install/iam
_policy.json
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  8759  100  8759    0     0  18286      0 --:--:-- --:--:-- --:--:-- 18286
[luis@vbox ~]$
```

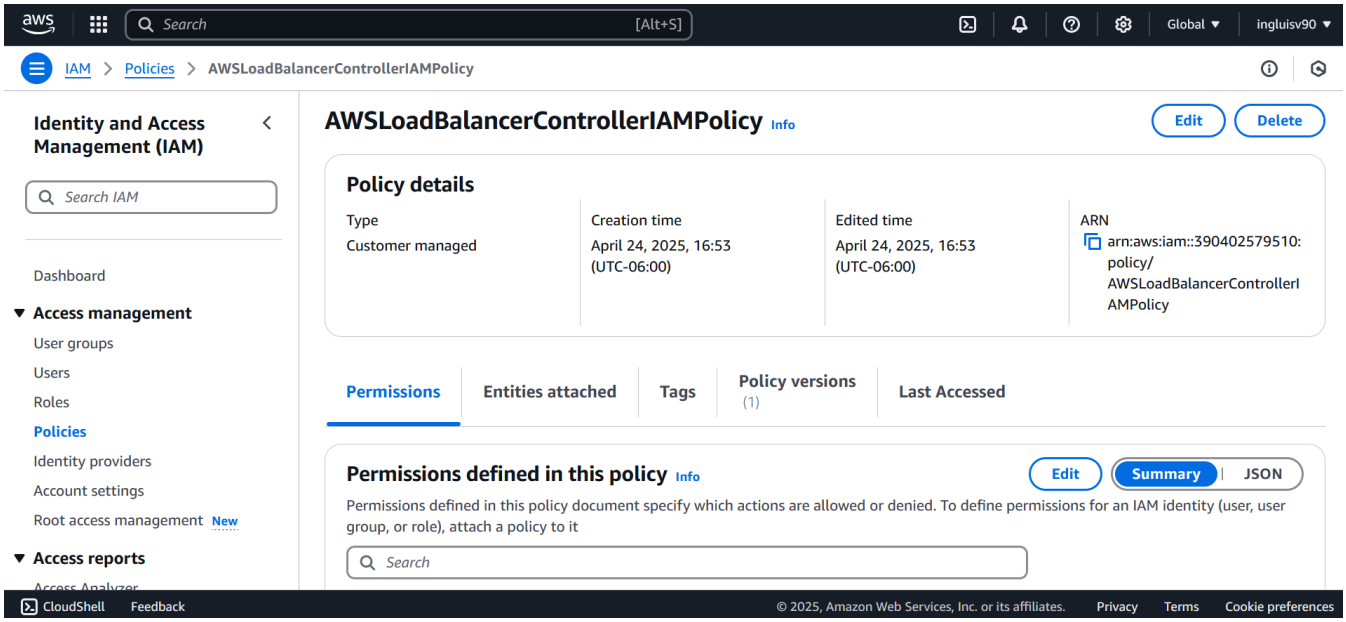
- Creating IAM policy

aws iam create-policy \

*--policy-name AWSLoadBalancerControllerIAMPolicy *
--policy-document file://iam_policy.json

```
[luis@vbox ~]$ aws iam create-policy \
    --policy-name AWSLoadBalancerControllerIAMPolicy \
    --policy-document file://iam_policy.json
{
    "Policy": {
        "PolicyName": "AWSLoadBalancerControllerIAMPolicy",
        "PolicyId": "ANPAVVZOOTQ3HQELVRTAM",
        "Arn": "arn:aws:iam::390402579510:policy/AWSLoadBalancerControllerIAMPolicy",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 0,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "CreateDate": "2025-04-24T22:53:06+00:00",
        "UpdateDate": "2025-04-24T22:53:06+00:00"
    }
}
[luis@vbox ~]$
```

on the AWS console

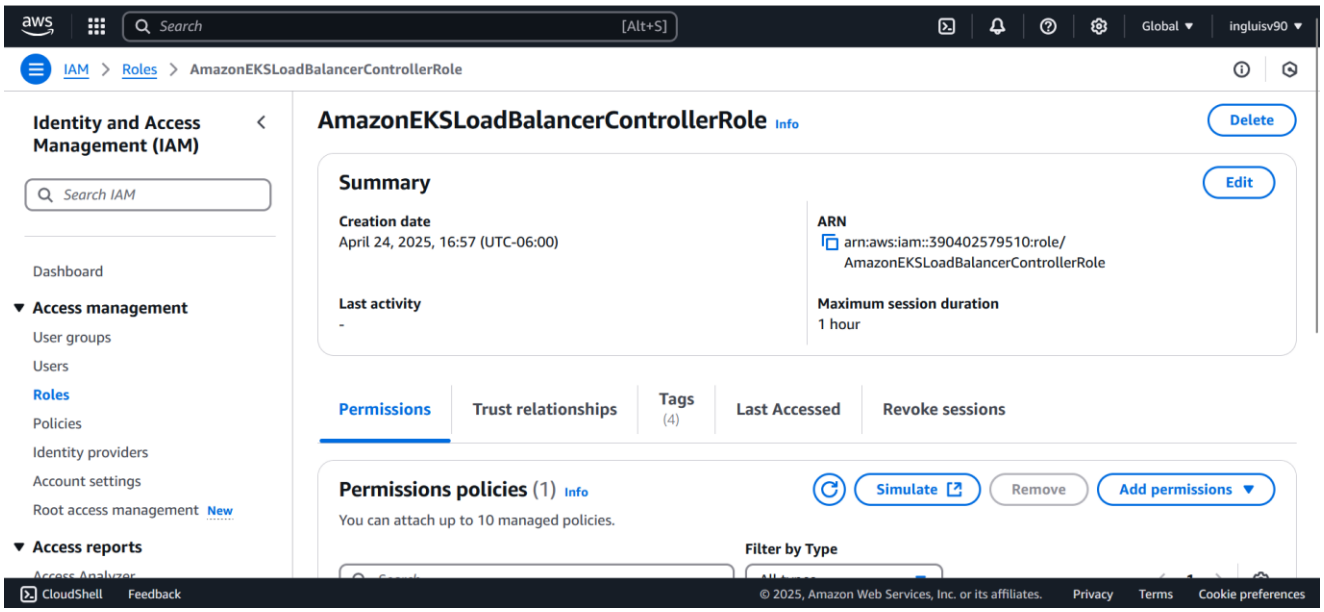


- Creating an IAM role, service account attached to the role

*eksctl create iamserviceaccount *
*--cluster=environment1eks *
*--namespace=kube-system *
*--name=aws-load-balancer-controller *
*--role-name AmazonEKSLoadBalancerControllerRole *
*--attach-policy-arn=arn:aws:iam::390402579510:policy/AWSLoadBalancerControllerIAMPolicy *
--approve

```
[luis@vbox ~]$ eksctl create iamserviceaccount \
  --cluster=environment1eks \
  --namespace=kube-system \
  --name=aws-load-balancer-controller \
  --role-name AmazonEKSLoadBalancerControllerRole \
  --attach-policy-arn=arn:aws:iam::390402579510:policy/AWSLoadBalancerControllerIAMPolicy \
  --approve
2025-04-24 16:57:48 [ℹ]  1 iamserviceaccount (kube-system/aws-load-balancer-controller) was included (based on the include/exc
lude rules)
2025-04-24 16:57:48 [!]  serviceaccounts that exist in Kubernetes will be excluded, use --override-existing-serviceaccounts to
 override
2025-04-24 16:57:48 [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",
    } }2025-04-24 16:57:48 [ℹ]  building iamserviceaccount stack "eksctl-environment1eks-addon-iamserviceaccount-kube-system-a
ws-load-balancer-controller"
2025-04-24 16:57:48 [ℹ]  deploying stack "eksctl-environment1eks-addon-iamserviceaccount-kube-system-aws-load-balancer-control
ler"
2025-04-24 16:57:48 [ℹ]  waiting for CloudFormation stack "eksctl-environment1eks-addon-iamserviceaccount-kube-system-aws-load
-balancer-controller"
2025-04-24 16:58:19 [ℹ]  waiting for CloudFormation stack "eksctl-environment1eks-addon-iamserviceaccount-kube-system-aws-load
-balancer-controller"
2025-04-24 16:58:19 [ℹ]  created serviceaccount "kube-system/aws-load-balancer-controller"
[luis@vbox ~]$
```

on the aws console



- Adding the repo for creating application load balancer with:

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

```
[luis@vbox ~]$ helm repo add eks https://aws.github.io/eks-charts
"eks" has been added to your repositories
[luis@vbox ~]$
```

- Installing aws balancer controller

*helm install aws-load-balancer-controller eks/aws-load-balancer-controller -n kube-system *
*--set clusterName=environment1eks *
*--set serviceAccount.create=false *
*--set serviceAccount.name=aws-load-balancer-controller *
*--set region=us-east-1 *
--set vpcId=vpc-0898b30b384bed685

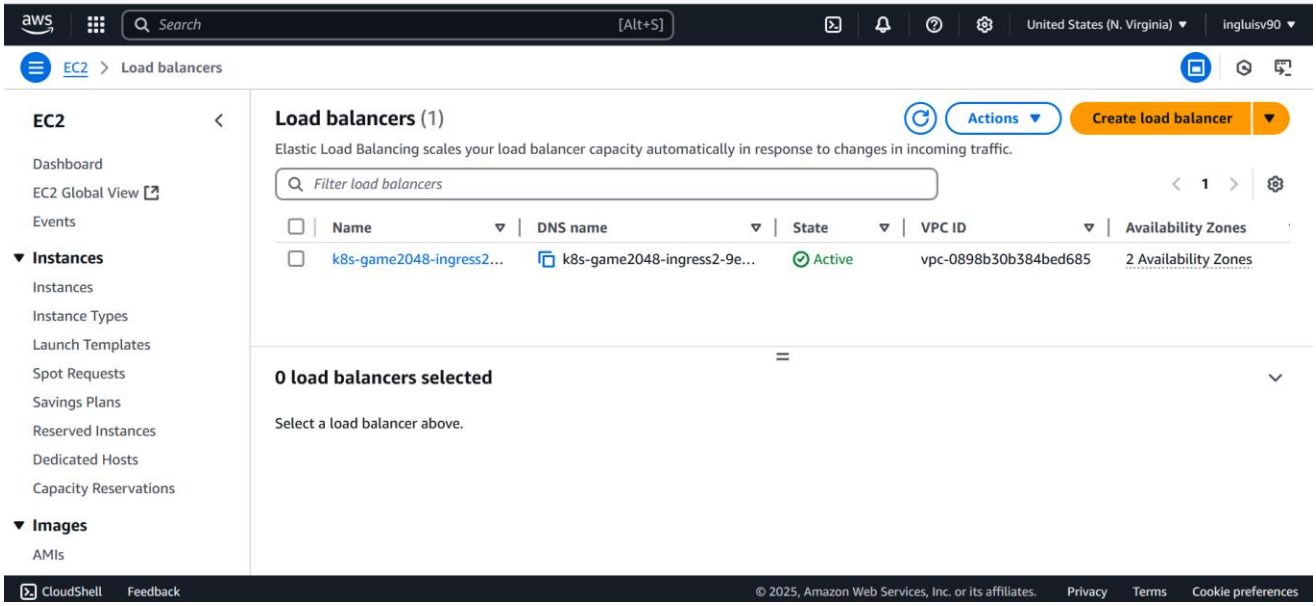
```
[luis@vbox ~]$ helm install aws-load-balancer-controller eks/aws-load-balancer-controller -n kube-system \
  --set clusterName=environment1eks \
  --set serviceAccount.create=false \
  --set serviceAccount.name=aws-load-balancer-controller \
  --set region=us-east-1 \
  --set vpcId=vpc-0898b30b384bed685
NAME: aws-load-balancer-controller
LAST DEPLOYED: Thu Apr 24 17:24:58 2025
NAMESPACE: kube-system
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
AWS Load Balancer controller installed!
[luis@vbox ~]$
```

Verifying that the deployments are running:

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

```
[luis@vbox ~]$ kubectl get deployment -n kube-system aws-load-balancer-controller
NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
aws-load-balancer-controller   2/2     2            2           103s
[luis@vbox ~]$
```

Checking the load balancer in the AWS Console:

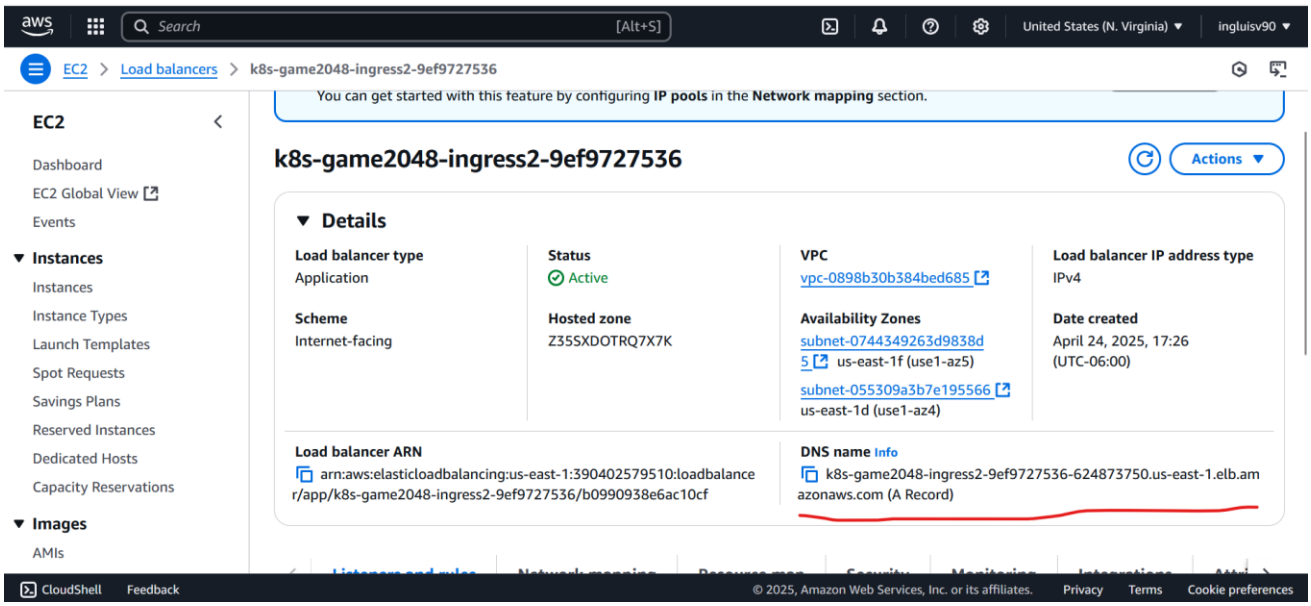


With the load balancer created we have now an address:

kubectl get ingress -n game-2048

```
[luis@vbox ~]$ kubectl get ingress -n game-2048
NAME           CLASS   HOSTS   ADDRESS                                                                  PORTS   AGE
ingress-2048   alb     *       k8s-game2048-ingress2-9ef9727536-624873750.us-east-1.elb.amazonaws.com   80      60m
[luis@vbox ~]$
```

Looking for the same address in the AWS console:



- Finally we have our application running

http://k8s-game2048-ingress2-9ef9727536-624873750.us-east-1.elb.amazonaws.com/