Kubernetes End to End Project on EKS(Amazon Kubernetes Service)

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.

Project Title: Deploying 2048 Game App on Amazon EKS

Project Description

A Kubernetes End-to-End (E2E) project for deploying a 2048 game app on Amazon Elastic Kubernetes Service (EKS) involves setting up, deploying, and managing the popular 2048 game application on a Kubernetes cluster running on AWS EKS. This project aims to demonstrate how to containerize a web application, deploy it on EKS, manage the cluster, and expose the application to users.

Containerization

I began by containerizing the 2048 game using Docker. This involved creating a Dockerfile to define the application's runtime environment and dependencies, ultimately resulting in a Docker image ready for deployment.

Amazon EKS Setup

I set up an Amazon EKS cluster, configuring the required resources and network settings using AWS services. This step included authentication and permissions setup to interact with the EKS cluster.

Deployment

The containerized 2048 game was deployed on the EKS cluster using Kubernetes. I defined Kubernetes deployment and service YAML files to ensure the application's efficient management and availability.

Scaling and Management

I explored Kubernetes's scaling capabilities, adjusting the number of application replicas based on demand. This ensured the game could handle varying levels of user traffic seamlessly.

Application Exposure

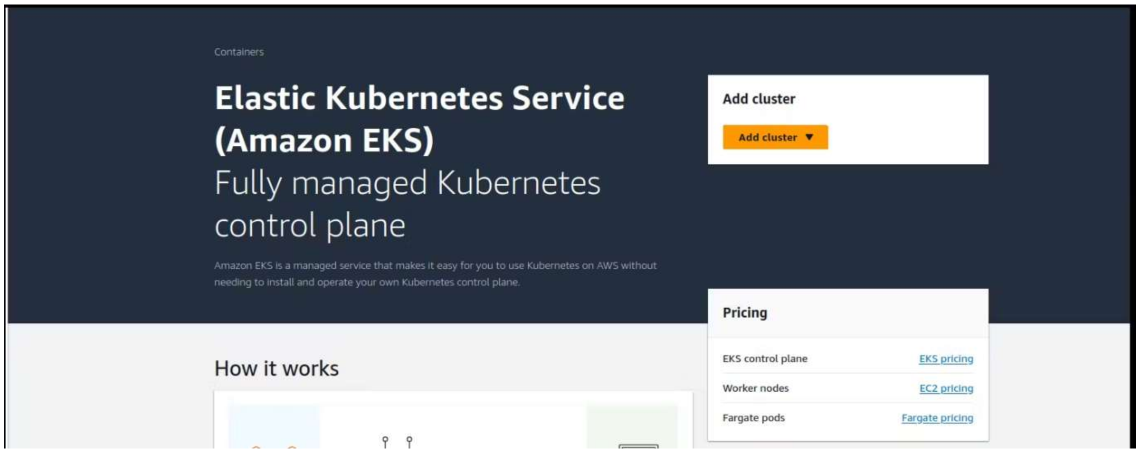
To make the 2048 game accessible to users, I created a Kubernetes service to expose it securely over the internet. Additionally, I could have implemented an Ingress controller for more advanced routing

Create IAM Roles

You need two IAM roles:

Cluster Role

1. Go to **IAM > Roles > Create Role**
2. Choose **EKS > EKS - Cluster**
3. Attach **AmazonEKSClusterPolicy**
4. Name: eks-cluster-role



Containers

Elastic Kubernetes Service (Amazon EKS)

Fully managed Kubernetes control plane

Amazon EKS is a managed service that makes it easy for you to use Kubernetes on AWS without needing to install and operate your own Kubernetes control plane.

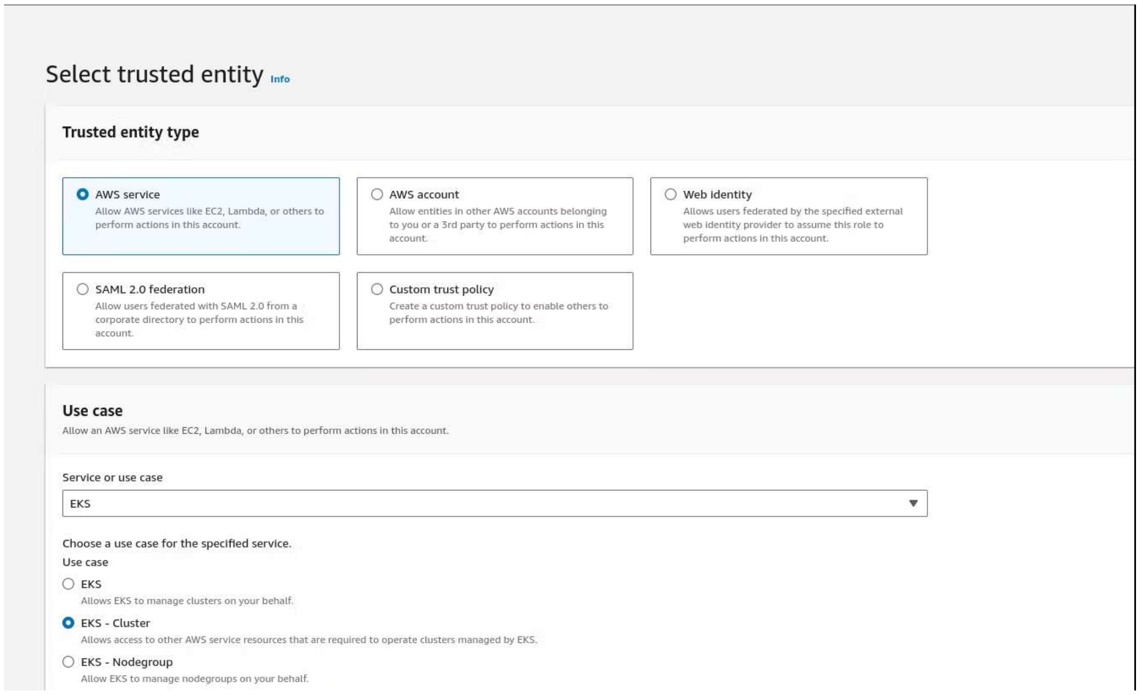
Add cluster

Add cluster ▼

Pricing

EKS control plane	EKS pricing
Worker nodes	EC2 pricing
Fargate pods	Fargate pricing

How it works



Select trusted entity Info

Trusted entity type

[✓ AWS service] Allow AWS services like EC2, Lambda, or others to perform actions in this account.

AWS account: Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

Web identity: Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

SAML 2.0 federation: Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.

Custom trust policy: Create a custom trust policy to enable others to perform actions in this account.

Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case: EKS

Choose a use case for the specified service.

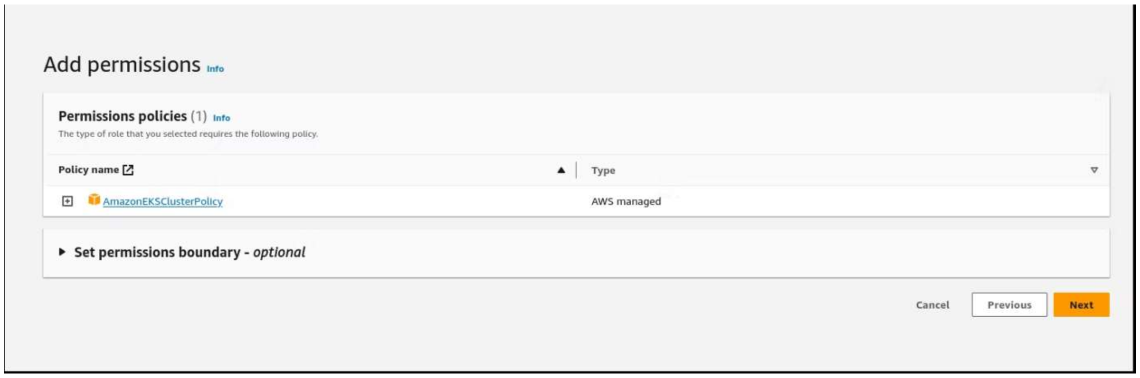
Use case

EKS: Allows EKS to manage clusters on your behalf.

[✓ EKS - Cluster] Allows access to other AWS service resources that are required to operate clusters managed by EKS.

EKS - Nodegroup: Allow EKS to manage nodegroups on your behalf.

Create an IAM role eks-cluster-role with 1 policy attached: AmazonEKSClusterPolicy


Add permissions Info

Permissions policies (1) Info

The type of role that you selected requires the following policy.

Policy name	Type
AmazonEKSClusterPolicy	AWS managed

▸ Set permissions boundary - *optional*

Create another IAM role 'eks-node-grp-role' with 3 policies attached:

(Allows EC2 instances to call AWS services on your behalf.)

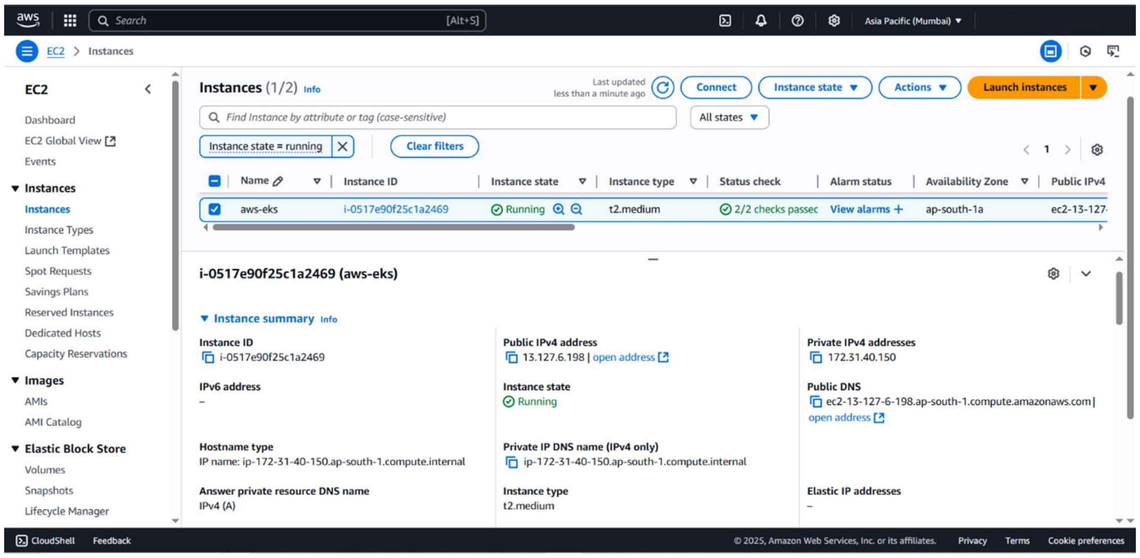
Node Group Role

1. Create another role
2. Choose EKS > EKS - Nodegroup
3. Attach the following policies:
    - AmazonEKSWorkerNodePolicy
    - AmazonEC2ContainerRegistryReadOnly
    - AmazonEKS_CNI_Policy
4. Name: eks-node-grp-role

Step-3: Launch EC2 Ubuntu 22.04 Instance (if not already done)

You can launch it from the AWS Console:

- AMI: Ubuntu 22.04
- Instance Type: t2.medium or t3.medium (recommended for this task)
- Enable auto-assign public IP
- Add a key pair (e.g.,awsdevops)



Instances (1/2) Info

Find Instance by attribute or tag (case-sensitive)

Instance state = running

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
aws-eks	i-0517e90f25c1a2469	Running	t2.medium	2/2 checks passec	View alarms +	ap-south-1a	ec2-13-127-

i-0517e90f25c1a2469 (aws-eks)

Instance summary Info

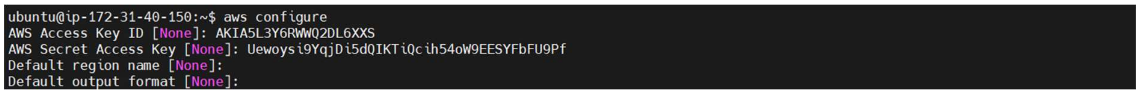
Instance ID i-0517e90f25c1a2469	Public IPv4 address 13.127.6.198 | open address	Private IPv4 addresses 172.31.40.150
IPv6 address –	Instance state Running	Public DNS ec2-13-127-6-198.ap-south-1.compute.amazonaws.com | open address
Hostname type IP name: ip-172-31-40-150.ap-south-1.compute.internal	Private IP DNS name (IPv4 only) ip-172-31-40-150.ap-south-1.compute.internal	
Answer private resource DNS name IPv4 (A)	Instance type t2.medium	Elastic IP addresses –

Connect to Your EC2 Ubuntu Instance

From your local terminal:

- sudo apt update
- sudo apt upgrade -y
- curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
- sudo apt install zip
- sudo apt install unzip -y
- unzip awscliv2.zip
- sudo ./aws/install
- aws –version

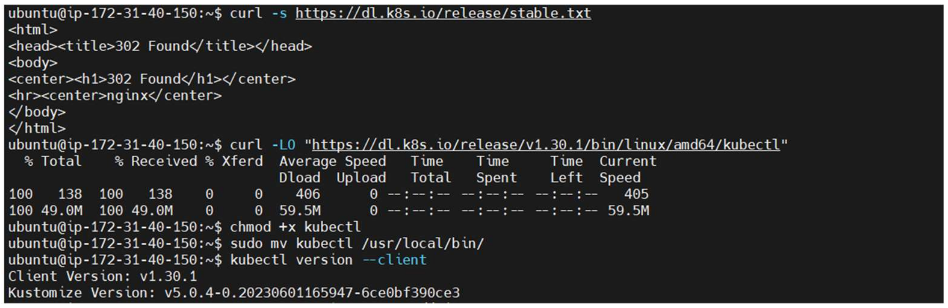
Configure AWS CLI:



```
ubuntu@ip-172-31-40-150:~$ aws configure
AWS Access Key ID [None]: AKIA5L3Y6RWWQ2DL6XXS
AWS Secret Access Key [None]: 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
Default region name [None]:
Default output format [None]:
```

Install kubectl

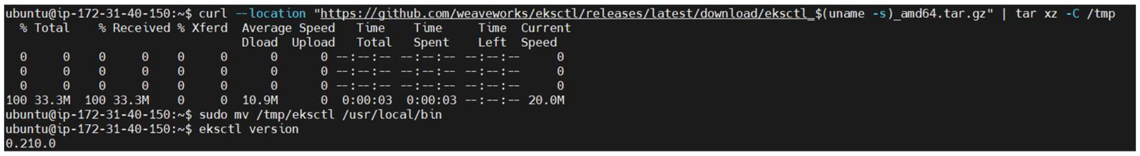
- curl -s https://dl.k8s.io/release/stable.txt
  Replace v1.30.1 with the version you got (or use the other current version)
- curl -LO https://dl.k8s.io/release/v1.30.1/bin/linux/amd64/kubectl
- chmod +x kubectl
- sudo mv kubectl /usr/local/bin/
- kubectl version –client



```
ubuntu@ip-172-31-40-150:~$ curl -s https://dl.k8s.io/release/stable.txt
<html>
<head><title>302 Found</title></head>
<body>
<center><h1>302 Found</h1></center>
<hr><center>nginx</center>
</body>
</html>
ubuntu@ip-172-31-40-150:~$ curl -LO "https://dl.k8s.io/release/v1.30.1/bin/linux/amd64/kubectl"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   138  100   138    0     0    406      0 --:--:-- --:--:-- --:--:--   405
100 49.0M  100 49.0M    0     0  59.5M      0 --:--:-- --:--:-- --:--:-- 59.5M
ubuntu@ip-172-31-40-150:~$ chmod +x kubectl
ubuntu@ip-172-31-40-150:~$ sudo mv kubectl /usr/local/bin/
ubuntu@ip-172-31-40-150:~$ kubectl version --client
Client Version: v1.30.1
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
```

Install eksctl

- curl –location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz
- sudo mv eksctl /usr/local/bin
- eksctl version



```
ubuntu@ip-172-31-40-150:~$ curl --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100 33.3M  100 33.3M    0     0  10.9M      0  0:00:03  0:00:03 --:--:-- 20.0M
ubuntu@ip-172-31-40-150:~$ sudo mv /tmp/eksctl /usr/local/bin
ubuntu@ip-172-31-40-150:~$ eksctl version
0.210.0
```

Create EKS Cluster:

```
eksctl create cluster \
--name 2048-eks-cluster \
--version 1.29 \
--region ap-south-1 \
--nodegroup-name 2048-node-group \
--node-type t3.medium \
--nodes 1 \
--nodes-min 1 \
--nodes-max 2 \
--node-volume-size 20 \
--managed \
--with-oidc \
--ssh-access \
--ssh-public-key awsdevops\
--full-ecr-access \
--asg-access \
--alb-ingress-access
```

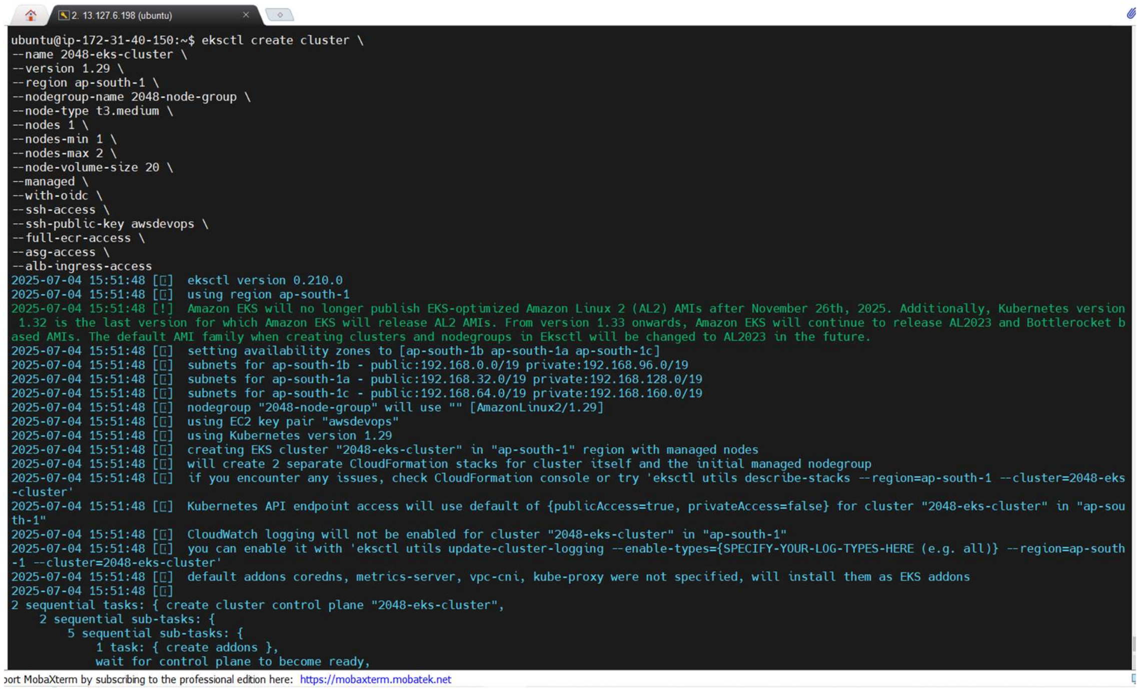
Note:

- Replace the region with your desired location you choose.
- Replace nodegroup-name according to your desired name.
- Replace (awsdevops) with the exact name of your EC2 key pair created in the AWS Console.

Verify Cluster Creation

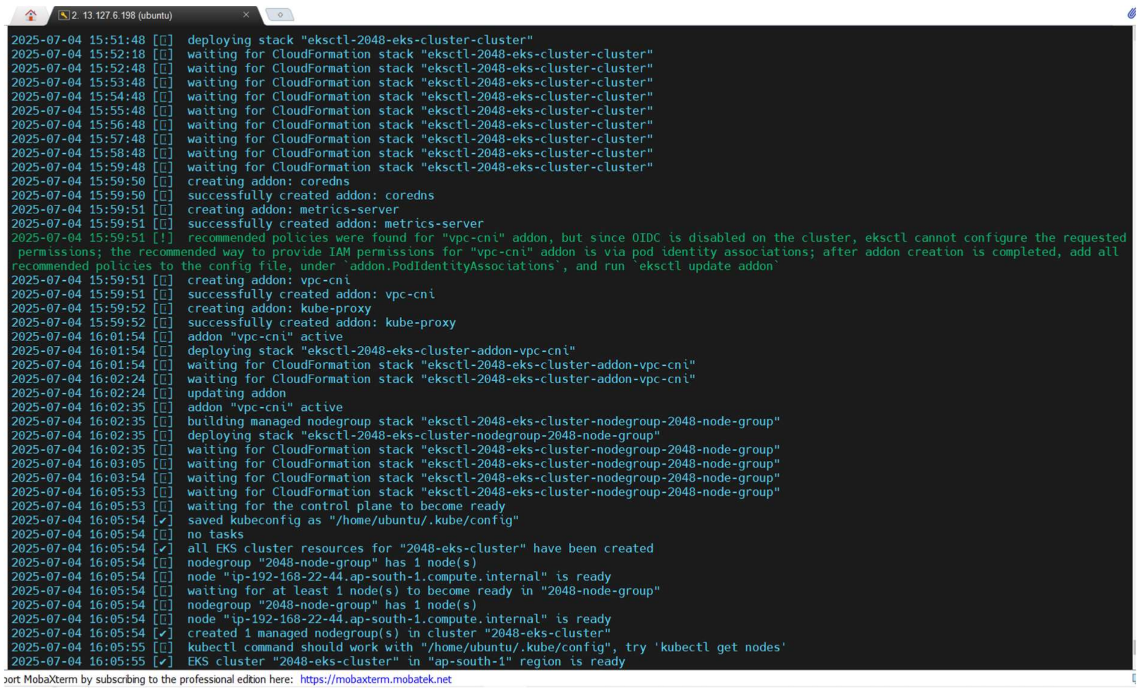
Cluster creation takes around 15 minutes. Once done:

- aws eks --region us-east-1 describe-cluster --name 2048-eks-cluster --query "cluster.status"



```
ubuntu@ip-172-31-40-150:~$ eksctl create cluster \
--name 2048-eks-cluster \
--version 1.29 \
--region ap-south-1 \
--nodegroup-name 2048-node-group \
--node-type t3.medium \
--nodes 1 \
--nodes-min 1 \
--nodes-max 2 \
--node-volume-size 20 \
--managed \
--with-oidc \
--ssh-access \
--ssh-public-key awsdevops \
--full-ecr-access \
--asg-access \
--alb-ingress-access
2025-07-04 15:51:48 [ℹ]  eksctl version 0.210.0
2025-07-04 15:51:48 [ℹ]  using region ap-south-1
2025-07-04 15:51:48 [!]  Amazon EKS will no longer publish EKS-optimized Amazon Linux 2 (AL2) AMIs after November 26th, 2025. Additionally, Kubernetes version
 1.32 is the last version for which Amazon EKS will release AL2 AMIs. From version 1.33 onwards, Amazon EKS will continue to release AL2023 and Bottlerocket b
ased AMIs. The default AMI family when creating clusters and nodegroups in Eksctl will be changed to AL2023 in the future.
2025-07-04 15:51:48 [ℹ]  setting availability zones to [ap-south-1b ap-south-1a ap-south-1c]
2025-07-04 15:51:48 [ℹ]  subnets for ap-south-1b - public:192.168.0.0/19 private:192.168.96.0/19
2025-07-04 15:51:48 [ℹ]  subnets for ap-south-1a - public:192.168.32.0/19 private:192.168.128.0/19
2025-07-04 15:51:48 [ℹ]  subnets for ap-south-1c - public:192.168.64.0/19 private:192.168.160.0/19
2025-07-04 15:51:48 [ℹ]  nodegroup "2048-node-group" will use "" [AmazonLinux2/1.29]
2025-07-04 15:51:48 [ℹ]  using EC2 key pair "awsdevops"
2025-07-04 15:51:48 [ℹ]  using Kubernetes version 1.29
2025-07-04 15:51:48 [ℹ]  creating EKS cluster "2048-eks-cluster" in "ap-south-1" region with managed nodes
2025-07-04 15:51:48 [ℹ]  will create 2 separate CloudFormation stacks for cluster itself and the initial managed nodegroup
2025-07-04 15:51:48 [ℹ]  if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=ap-south-1 --cluster=2048-eks
-cluster'
2025-07-04 15:51:48 [ℹ]  Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "2048-eks-cluster" in "ap-sou
th-1"
2025-07-04 15:51:48 [ℹ]  CloudWatch logging will not be enabled for cluster "2048-eks-cluster" in "ap-south-1"
2025-07-04 15:51:48 [ℹ]  you can enable it with 'eksctl utils update-cluster-logging --enable-types={SPECIFY-YOUR-LOG-TYPES-HERE (e.g. all)} --region=ap-south
-1 --cluster=2048-eks-cluster'
2025-07-04 15:51:48 [ℹ]  default addons coredns, metrics-server, vpc-cni, kube-proxy were not specified, will install them as EKS addons
2025-07-04 15:51:48 [ℹ]
2 sequential tasks: { create cluster control plane "2048-eks-cluster",
    2 sequential sub-tasks: {
        5 sequential sub-tasks: {
            1 task: { create addons },
            wait for control plane to become ready,
```

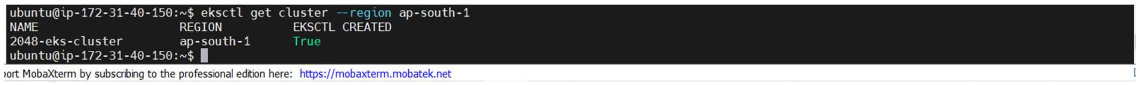
ort MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net



```
2025-07-04 15:51:48 [ℹ]  deploying stack "eksctl-2048-eks-cluster-cluster"
2025-07-04 15:52:18 [ℹ]  waiting for CloudFormation stack "eksctl-2048-eks-cluster-cluster"
2025-07-04 15:52:48 [ℹ]  waiting for CloudFormation stack "eksctl-2048-eks-cluster-cluster"
2025-07-04 15:53:48 [ℹ]  waiting for CloudFormation stack "eksctl-2048-eks-cluster-cluster"
2025-07-04 15:54:48 [ℹ]  waiting for CloudFormation stack "eksctl-2048-eks-cluster-cluster"
2025-07-04 15:55:48 [ℹ]  waiting for CloudFormation stack "eksctl-2048-eks-cluster-cluster"
2025-07-04 15:56:48 [ℹ]  waiting for CloudFormation stack "eksctl-2048-eks-cluster-cluster"
2025-07-04 15:57:48 [ℹ]  waiting for CloudFormation stack "eksctl-2048-eks-cluster-cluster"
2025-07-04 15:58:48 [ℹ]  waiting for CloudFormation stack "eksctl-2048-eks-cluster-cluster"
2025-07-04 15:59:48 [ℹ]  waiting for CloudFormation stack "eksctl-2048-eks-cluster-cluster"
2025-07-04 15:59:50 [ℹ]  creating addon: coredns
2025-07-04 15:59:50 [ℹ]  successfully created addon: coredns
2025-07-04 15:59:51 [ℹ]  creating addon: metrics-server
2025-07-04 15:59:51 [ℹ]  successfully created addon: metrics-server
2025-07-04 15:59:51 [!]  recommended policies were found for "vpc-cni" addon, but since OIDC is disabled on the cluster, eksctl cannot configure the requested
 permissions; the recommended way to provide IAM permissions for "vpc-cni" addon is via pod identity associations; after addon creation is completed, add all
recommended policies to the config file, under `addon.PodIdentityAssociations`, and run `eksctl update addon`
2025-07-04 15:59:51 [ℹ]  creating addon: vpc-cni
2025-07-04 15:59:51 [ℹ]  successfully created addon: vpc-cni
2025-07-04 15:59:52 [ℹ]  creating addon: kube-proxy
2025-07-04 15:59:52 [ℹ]  successfully created addon: kube-proxy
2025-07-04 16:01:54 [ℹ]  addon "vpc-cni" active
2025-07-04 16:01:54 [ℹ]  deploying stack "eksctl-2048-eks-cluster-addon-vpc-cni"
2025-07-04 16:01:54 [ℹ]  waiting for CloudFormation stack "eksctl-2048-eks-cluster-addon-vpc-cni"
2025-07-04 16:02:24 [ℹ]  waiting for CloudFormation stack "eksctl-2048-eks-cluster-addon-vpc-cni"
2025-07-04 16:02:24 [ℹ]  updating addon
2025-07-04 16:02:35 [ℹ]  addon "vpc-cni" active
2025-07-04 16:02:35 [ℹ]  building managed nodegroup stack "eksctl-2048-eks-cluster-nodegroup-2048-node-group"
2025-07-04 16:02:35 [ℹ]  deploying stack "eksctl-2048-eks-cluster-nodegroup-2048-node-group"
2025-07-04 16:02:35 [ℹ]  waiting for CloudFormation stack "eksctl-2048-eks-cluster-nodegroup-2048-node-group"
2025-07-04 16:03:05 [ℹ]  waiting for CloudFormation stack "eksctl-2048-eks-cluster-nodegroup-2048-node-group"
2025-07-04 16:03:54 [ℹ]  waiting for CloudFormation stack "eksctl-2048-eks-cluster-nodegroup-2048-node-group"
2025-07-04 16:05:53 [ℹ]  waiting for CloudFormation stack "eksctl-2048-eks-cluster-nodegroup-2048-node-group"
2025-07-04 16:05:53 [ℹ]  waiting for the control plane to become ready
2025-07-04 16:05:54 [✔]  saved kubeconfig as "/home/ubuntu/.kube/config"
2025-07-04 16:05:54 [ℹ]  no tasks
2025-07-04 16:05:54 [✔]  all EKS cluster resources for "2048-eks-cluster" have been created
2025-07-04 16:05:54 [ℹ]  nodegroup "2048-node-group" has 1 node(s)
2025-07-04 16:05:54 [ℹ]  node "ip-192-168-22-44.ap-south-1.compute.internal" is ready
2025-07-04 16:05:54 [ℹ]  waiting for at least 1 node(s) to become ready in "2048-node-group"
2025-07-04 16:05:54 [ℹ]  nodegroup "2048-node-group" has 1 node(s)
2025-07-04 16:05:54 [ℹ]  node "ip-192-168-22-44.ap-south-1.compute.internal" is ready
2025-07-04 16:05:54 [✔]  created 1 managed nodegroup(s) in cluster "2048-eks-cluster"
2025-07-04 16:05:55 [ℹ]  kubectl command should work with "/home/ubuntu/.kube/config", try 'kubectl get nodes'
2025-07-04 16:05:55 [✔]  EKS cluster "2048-eks-cluster" in "ap-south-1" region is ready
```

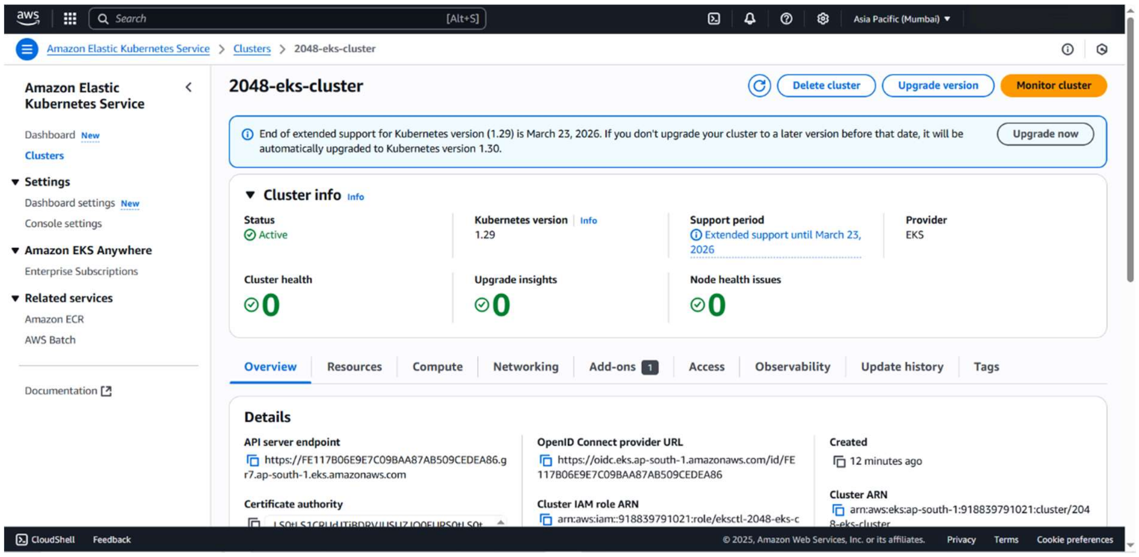
ort MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

- eksctl get cluster --region us-east-1



```
ubuntu@ip-172-31-40-150:~$ eksctl get cluster --region ap-south-1
NAME                    REGION          EKSCTL CREATED
2048-eks-cluster        ap-south-1      True
ubuntu@ip-172-31-40-150:~$
```

ort MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net



2048-eks-cluster

End of extended support for Kubernetes version (1.29) is March 23, 2026. If you don't upgrade your cluster to a later version before that date, it will be automatically upgraded to Kubernetes version 1.30.

Cluster info

Status	Kubernetes version	Support period	Provider
Active	1.29	Extended support until March 23, 2026	EKS

Cluster health	Upgrade insights	Node health issues
0	0	0

Details

API server endpoint: https://FE117B06E9E7C09BAA87AB509CEDEA86.gr7.ap-south-1.eks.amazonaws.com

OpenID Connect provider URL: https://oidc.eks.ap-south-1.amazonaws.com/id/FE117B06E9E7C09BAA87AB509CEDEA86

Created: 12 minutes ago

Certificate authority

Cluster IAM role ARN: arn:aws:iam::918839791021:role/eksctl-2048-eks-c

Cluster ARN: arn:aws:eks:ap-south-1:918839791021:cluster/2048-eks-cluster

Authentication & Connection:

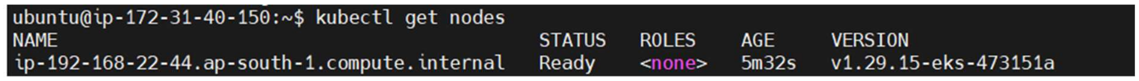
Update kubeconfig

- aws eks update-kubeconfig --region ap-south-1 --name bhargav-eks-cluster

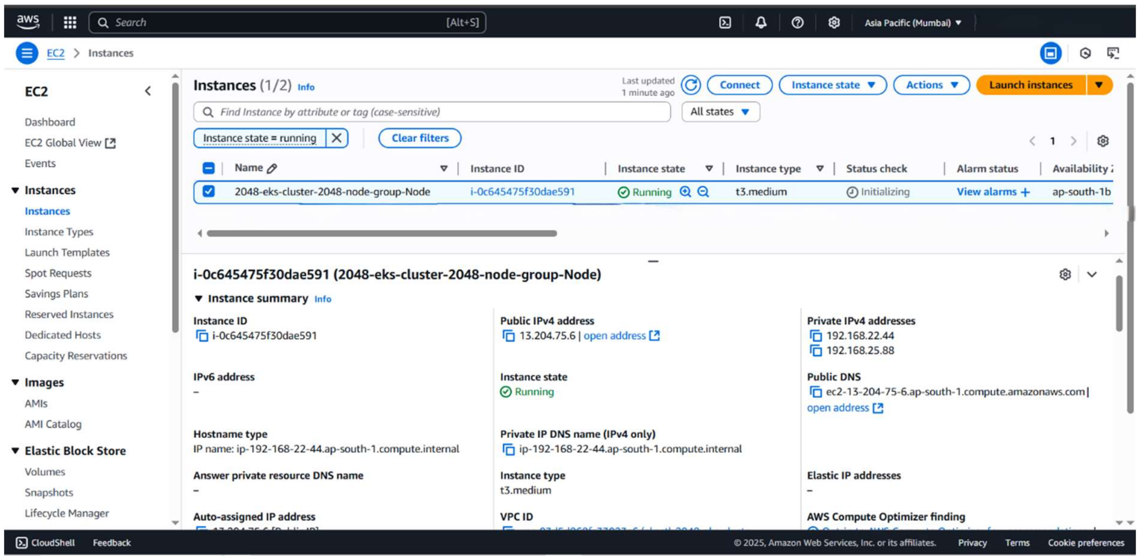
Verify Cluster

- kubectl get nodes

You should see a node (worker node) in Ready state.



```
ubuntu@ip-172-31-40-150:~$ kubectl get nodes
NAME                                        STATUS   ROLES    AGE     VERSION
ip-192-168-22-44.ap-south-1.compute.internal   Ready    <none>   5m32s   v1.29.15-eks-473151a
```



Instances (1/2)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
2048-eks-cluster-2048-node-group-Node	i-0c645475f30dae591	Running	t3.medium	Initializing	View alarms +	ap-south-1b

i-0c645475f30dae591 (2048-eks-cluster-2048-node-group-Node)

Instance summary

Instance ID: i-0c645475f30dae591

Public IPv4 address: 13.204.75.6 | open address

Private IPv4 addresses: 192.168.22.44, 192.168.25.88

IPv6 address: –

Instance state: Running

Public DNS: ec2-13-204-75-6.ap-south-1.compute.amazonaws.com | open address

Hostname type: IP name: ip-192-168-22-44.ap-south-1.compute.internal

Private IP DNS name (IPv4 only): ip-192-168-22-44.ap-south-1.compute.internal

Answer private resource DNS name: –

Instance type: t3.medium

Elastic IP addresses: –

Auto-assigned IP address

VPC ID

AWS Compute Optimizer finding

Create and Deploy Pod:

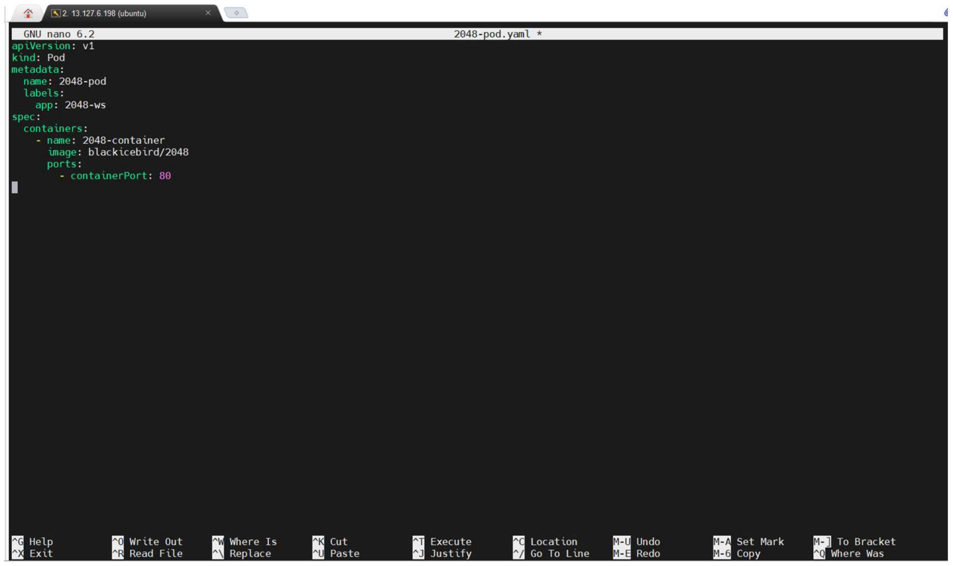
Create the YAML Manifest File (2048-pod.yaml):

Note: Use any text editor to create the YAML file. If using AWS CloudShell or a Linux terminal:

- nano 2048-pod.yaml

Paste the following YAML code inside the file:

```yaml
apiVersion: v1

kind: Pod

metadata:

  name: 2048-pod

  labels:

    app: 2048-ws

spec:

  containers:

    - name: 2048-container

      image: blackicebird/2048

      ports:

        - containerPort: 80
```

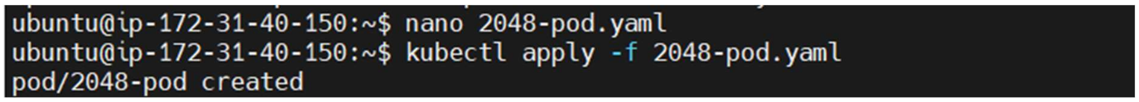


```
  GNU nano 6.2                         2048-pod.yaml *
apiVersion: v1
kind: Pod
metadata:
  name: 2048-pod
  labels:
    app: 2048-ws
spec:
  containers:
    - name: 2048-container
      image: blackicebird/2048
      ports:
        - containerPort: 80


^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark  M-] To Bracket
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line M-E Redo      M-6 Copy      ^Q Where Was
```

Press CTRL + O → hit Enter to save

Press CTRL + X to exit nano

- kubectl apply -f 2048-pod.yaml
- kubectl get pods



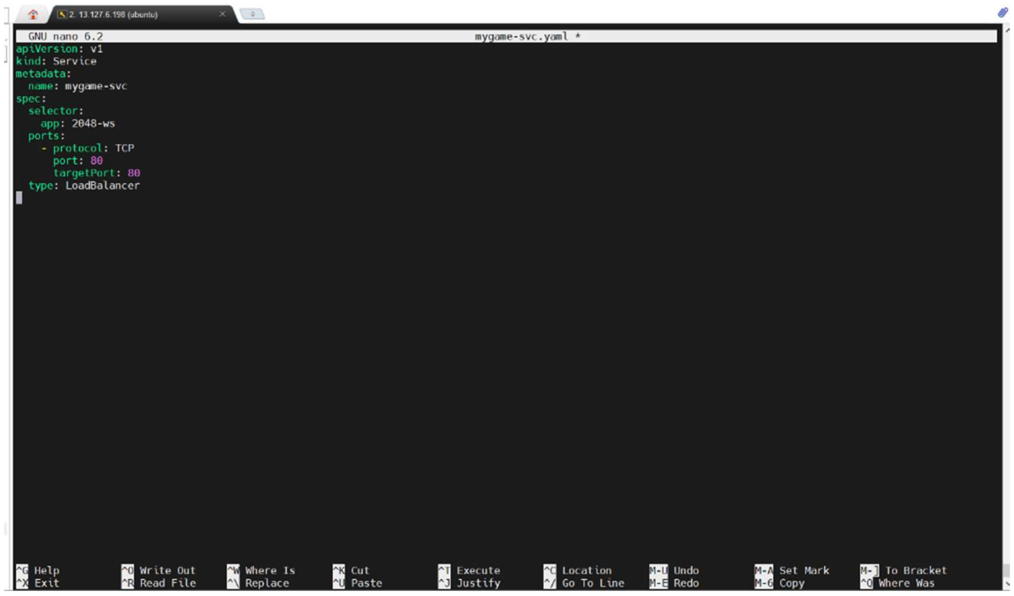
```
ubuntu@ip-172-31-40-150:~$ nano 2048-pod.yaml
ubuntu@ip-172-31-40-150:~$ kubectl apply -f 2048-pod.yaml
pod/2048-pod created
```

Expose Pod with LoadBalancer:

Create Service (mygame-svc.yaml):

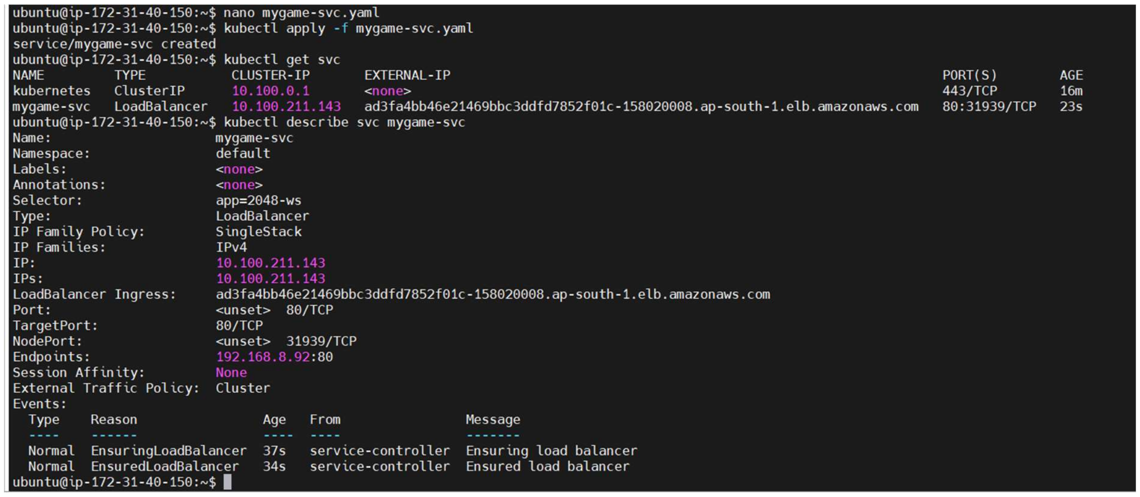
- nano mygame-svc.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: mygame-svc
spec:
  selector:
    app: 2048-ws
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: LoadBalancer
```



```
apiVersion: v1
kind: Service
metadata:
  name: mygame-svc
spec:
  selector:
    app: 2048-ws
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: LoadBalancer
```

- kubectl apply -f mygame-svc.yaml
- kubectl get svc
- kubectl describe svc mygame-svc



```
ubuntu@ip-172-31-40-150:~$ nano mygame-svc.yaml
ubuntu@ip-172-31-40-150:~$ kubectl apply -f mygame-svc.yaml
service/mygame-svc created
ubuntu@ip-172-31-40-150:~$ kubectl get svc
NAME         TYPE           CLUSTER-IP       EXTERNAL-IP                                                                PORT(S)        AGE
kubernetes   ClusterIP      10.100.0.1       <none>                                                                     443/TCP        16m
mygame-svc   LoadBalancer   10.100.211.143   ad3fa4bb46e21469bbc3ddfd7852f01c-158020008.ap-south-1.elb.amazonaws.com   80:31939/TCP   23s
ubuntu@ip-172-31-40-150:~$ kubectl describe svc mygame-svc
Name:                     mygame-svc
Namespace:                default
Labels:                   <none>
Annotations:              <none>
Selector:                 app=2048-ws
Type:                     LoadBalancer
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.100.211.143
IPs:                      10.100.211.143
LoadBalancer Ingress:     ad3fa4bb46e21469bbc3ddfd7852f01c-158020008.ap-south-1.elb.amazonaws.com
Port:                     <unset>  80/TCP
TargetPort:               80/TCP
NodePort:                 <unset>  31939/TCP
Endpoints:                192.168.8.92:80
Session Affinity:         None
External Traffic Policy:  Cluster
Events:
  Type    Reason                Age   From                Message
  ----    ------                ----  ----                -------
  Normal  EnsuringLoadBalancer  37s   service-controller  Ensuring load balancer
  Normal  EnsuredLoadBalancer   34s   service-controller  Ensured load balancer
ubuntu@ip-172-31-40-150:~$
```

Access the Application:

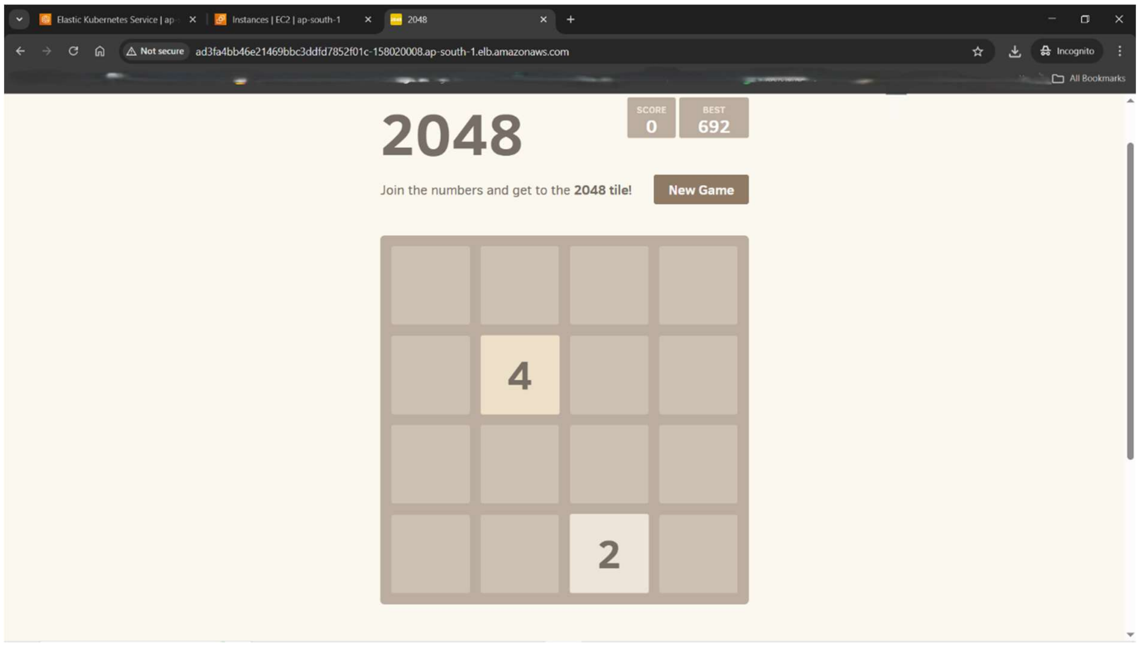
- Find the EXTERNAL-IP / ELB DNS Name:
- When you run the kubectl get svc command copy the External-IP of the load balancer.
- Your **EXTERNAL-IP** is the **ELB DNS name** assigned by AWS may look like this.

ad3fa4bb46e21469bbc3ddfd7852101c 158020008.ap south 1.elb.amazonaws.com

Access via Browser

Open this URL in your browser:

http:// ad3fa4bb46e21469bbc3ddfd7852101c 158020008.ap south 1.elb.amazonaws.com


Elastic Kubernetes Service | ap
Instances | EC2 | ap-south-1
2048
Not secure
ad3fa4bb46e21469bbc3ddfd7852f01c-158020008.ap-south-1.elb.amazonaws.com
Incognito
All Bookmarks
2048
SCORE
0
BEST
692
Join the numbers and get to the 2048 tile!
New Game
4
2