### Step01:- Install and configure AWS CLI

1.For the current version of the AWS CLI, use the following command:

**curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"**

For a specific version of the AWS CLI, append a hyphen and the version number to the filename. For this example the filename for version 2.0.30 would be awscli-exe-linux-x86\_64-2.0.30.zip resulting in the following command:

**curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64-2.0.30.zip" -o "awscliv2.zip"**

for reference please follow below link

<https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2-linux.html>

2. Now unzip the installer

If unzip is not available in your system please install it using

**sudo apt install unzip**

**unzip awscliv2.zip**

3.Run the install program. The installation command uses a file named install in the newly unzipped aws directory. By default, the files are all installed to /usr/local/aws-cli, and a symbolic link is created in /usr/local/bin. The command includes sudo to grant write permissions to those directories

**sudo ./aws/install**

4. Confirm the installation.

**aws --version**

### Step02: Configure AWS Command Line using Security Credentials

* Go to AWS Management Console --> Services --> IAM
* Select the IAM User: anis

If you don’t have IAM user created lets create it first because Never ever use Root User. (Highly not recommended) which is by default in creditionals

* Click on **Security credentials** tab
* Click on **Create access key**
* Copy Access ID and Secret access key
* Go to command line and provide the required detail

**aws configure**

**aws configure**

AWS Access Key ID [None]:- paste your access key here

AWS Secret Access Key [None]:- paste your secret key

Default region name [None]: us-east-1 choose as per your region

Default output format [None]: json

After that check your AWS CLI working or not

**aws ec2 describe-vpcs**

### Step-03: Install kubectl CLI

* **IMPORTANT NOTE:** Kubectl binaries for EKS please prefer to use from Amazon (**Amazon EKS-vended kubectl binary**)
* This will help us to get the exact Kubectl client version based on our EKS Cluster version. You can use the below documentation link to download the binary.
* Reference: https://docs.aws.amazon.com/eks/latest/userguide/install-kubectl.html

**1.**To install kubectl on Linux follow commands

So in this case ill take kubernetes 1.18 version if you want to take some other version follow reference link which will provide on start

**curl -o kubectl https://s3.us-west-2.amazonaws.com/amazon-eks/1.22.6/2022-03-09/bin/linux/amd64/kubectl**

**2.**now we will Download the SHA-256 sum for your cluster's Kubernetes version for Linux

**curl -o kubectl.sha256 https://s3.us-west-2.amazonaws.com/amazon-eks/1.22.6/2022-03-09/bin/linux/amd64/kubectl.sha256**

**3.** Now Check the SHA-256 sum for your downloaded binary**.**

**openssl sha1 -sha256 kubectl**

Compare the generated SHA-256 sum in the command output against your downloaded SHA-256 file. The two should match

**4.** Apply execute permissions to the binary**.**

**chmod +x ./kubectl**

Copy the binary to a folder in your PATH. If you have already installed a version of kubectl , then we recommend creating a $HOME/bin/kubectl and ensuring that $HOME/bin comes first in your $PATH.

**mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl&& export PATH=$PATH:$HOME/bin**

**5.** Add the $HOME/bin path to your shell initialization file so that it is configured when you open a shell.

**echo 'export PATH=$PATH:$HOME/bin' >> ~/.bashrc**

you can check with nano .bashrc so export path will shown in configuration

**6.** After you install kubectl , you can verify its version with the following command:

**kubectl version --short –client**

## **Step-04: Install eksctl CLI**

To install or upgrade eksctl on Linux using curl

For reference follow this link

https://docs.aws.amazon.com/eks/latest/userguide/eksctl.html

1. Download and extract the latest release of eksctl with the following command.

**curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl\_$(uname -s)\_amd64.tar.gz" | tar xz -C /tmp**

1. Move the extracted binary to /usr/local/bin.

**sudo mv /tmp/eksctl /usr/local/bin**

1. Test that your installation was successful with the following command

**eksctl version**

# **Create EKS Cluster & Node Groups**

## Step-01: Create EKS Cluster using eksctl

* Please note that It will take 15 to 20 minutes to create the Cluster Control Plane so be patience

**eksctl create cluster --name=irfan \**

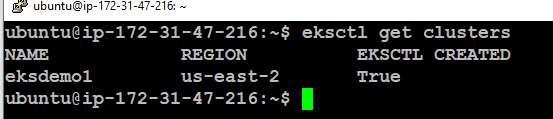
**--region=us-east-1 \**

**--zones=us-east-1a,us-east-1b \**

**--without-nodegroup**

After the cluster is ready then check with following commands

**eksctl get clusters**

****

## **Step-05: Create & Associate IAM OIDC Provider for our EKS Cluster**

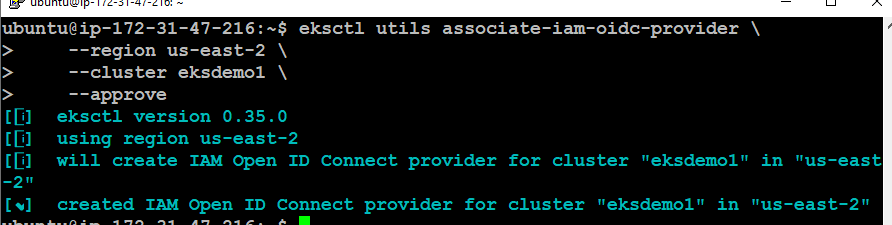
* To enable and use AWS IAM roles for Kubernetes service accounts on our EKS cluster, we must create & associate OIDC identity provider.
* To do so using eksctl we can use the below command.
* Use latest eksctl version (as on today the latest version is 0.35.0)

**eksctl utils associate-iam-oidc-provider \**

**--region us-east-1 \**

**--cluster irfan \**

**--approve**

****

## **Step-07: Create Node Group with additional Add-Ons in Public Subnets**

* These add-ons will create the respective IAM policies for us automatically within our Node Group role.
* # Create Public Node Group

eksctl create nodegroup --cluster=irfan \

--region=us-east-1 \

--name=eksdemo1-ng-public1 \

--node-type=t2.large \

--nodes=2 \

--nodes-min=2 \

--nodes-max=4 \

--node-volume-size=20 \

--ssh-access \

--ssh-public-key=myekskey \

--managed \

--asg-access \

--external-dns-access \

--full-ecr-access \

--appmesh-access \

--alb-ingress-access

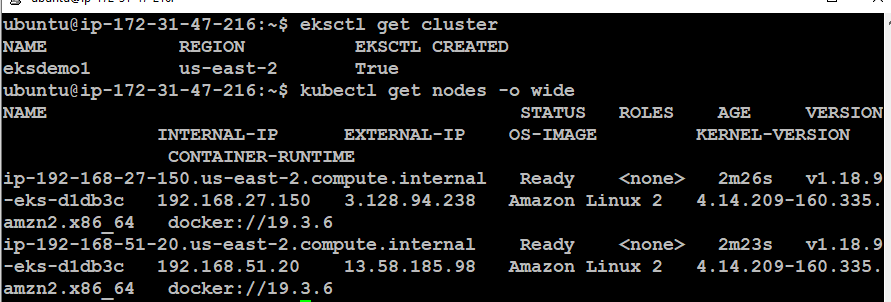
### # List Worker Nodes

# List EKS clusters

**eksctl get cluster**

# List Nodes in current kubernetes cluster

**kubectl get nodes -o wide**

****

**To deploy the AWS Load Balancer Controller to an Amazon EKS cluster**

# Create an IAM policy.

**curl -o iam\_policy.json https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.4.4/docs/install/iam\_policy.json**

### # Create an IAM policy using the policy downloaded in the previous step.

**aws iam create-policy \**

**--policy-name AWSLoadBalancerControllerIAMPolicy \**

**--policy-document** [**file://iam\_policy.json**](file://iam_policy.json)

**# Create** an **IAM role. Create a Kubernetes service account named aws-load-balancer-controller in the kube-system namespace for the AWS Load Balancer Controller and annotate the Kubernetes service account with the name of the IAM role.**

* Replace aws account number

**eksctl create iamserviceaccount \**

**--cluster=irfan \**

**--namespace=kube-system \**

**--name=aws-load-balancer-controller \**

**--role-name "AmazonEKSLoadBalancerControllerRole" \**

**--attach-policy-arn=arn:aws:iam::130129353970:policy/AWSLoadBalancerControllerIAMPolicy \**

**--approve**

**To check service on aws account run below command**

**Kubectl get sa -n kube-system**

# install helm

**curl -fsSL -o get\_helm.sh** [**https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3**](https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3)

**chmod 700 get\_helm.sh**

**./get\_helm.sh**

* Add the eks-charts repository.

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

* Update your local repo to make sure that you have the most recent charts.

**helm repo update**

# Install the AWS Load Balancer Controller using helm

* Replace cluster name

**helm install aws-load-balancer-controller eks/aws-load-balancer-controller \**

**-n kube-system \**

**--set clusterName=irfan \**

**--set serviceAccount.create=false \**

**--set serviceAccount.name=aws-load-balancer-controller**

* Verify that the controller is installed.

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

**Also Check for POD running by below command**

**kubectl get po -n kube-system**

**Also check logs of our pod-**

**Kubectl logs (pod id) -n kube-system ( check if any error found in logs)**

## Deploy a sample application

**Prerequisites**

* At least one public or private subnet in your cluster VPC.
* Have the AWS Load Balancer Controller deployed on your cluster.

# Deploy the game [2048](https://play2048.co/) as a sample application to verify that the AWS Load Balancer Controller creates an AWS ALB as a result of the ingress object.

* Download the manifest.

**curl -o 2048\_full.yaml https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.4.4/docs/examples/2048/2048\_full.yaml**

* Replace replicas to 1
* delete all below apiversion: networking.k8s.io/v1 kind: Ingress
* Open visual code studio and make one file abl.yaml and copy paste below content in that file :
* Add this content in yaml file –

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

alb.ingress.kubernetes.io/ssl-redirect: '443'

alb.ingress.kubernetes.io/certificate-arn: arn:aws:acm:us-east-1:180789647333:certificate/d86de939-8ffd-410f-adce-0ce1f5be6e0d

alb.ingress.kubernetes.io/listen-ports: '[{"HTTPS":443}, {"HTTP":80}]'

spec:

ingressClassName: alb

rules:

- host: game.solankiempire.site

http:

paths:

- path: /

pathType: Prefix

backend:

service:

name: service-2048

port:

number: 80

\*\*\* Note- Dot forget to change your arn ID which is taken from certificate manager in AWS.

* Apply the manifest to your cluster.

**kubectl apply -f 2048\_full.yaml**

**Check name space by applying below command (In output our game 2048 active will seen)**

**kubectl get ns**

**Now run below command to check our game running –**

**kubectl get all -n game-2048**

**Now make one yaml file with name ingress.yaml**

**vi ingress.yaml (Enter)**

**Now paste all content from visual code alb.yaml file to this ingress.yaml file.**

**Now Apply this file by :**

**kubectl apply -f ingress.yaml**

**( Ensure that output show ingress.networking.k8s.io/ingress-2048 created )**

* To verify the ingree type this –

**kubectl get ingress –A**

**Now check our load balancer status in EC2 (Either active or provisioning)**

**Now take DNS from load balancer and check in browser.**

you will get error of 404 because you didn’t mention default backend in ingress yaml file

open ingress yaml file with vi editer and add default backend below ingress classname

Now add defaultBackend content into ingress.yaml file

1st add default backend script into visual code studio alb.yaml file

Add default backend below Ingress class

defaultBackend:

**service:**

**name: app3-nginx-nodeport-service**

**port:**

**number: 80**

**After that change service : service-2048**

**Run kubectl apply -f ingress.yaml**

**Run kubectl get ingress –A**

**Again check dns in browser and game will reflect on browser.**

* verify that the ingress resource was created with the following command.

**kubectl get ingress/ingress-2048 -n game-2048**

# create one yaml file for ingress annotation and etc.

**Vi ingress.yaml**

* now take the dns address of your loadbalancer from AWS
* open url and paste this
* now take the dns address of your loadbalancer from AWS
* open url and paste this
* now you can access and play 2048-game 😊

------------------------------------------------------------------------------------------------------------------------------