Elastic Kubernetes Service (EKS) Application Deployment

1. Firstly, deploy a EKS Cluster & node group with the required set of policies and role permissions using Terraform/ console. And then create a new AWS Linux EC2 or Cloud Shell and login in.
2. As it’s an AWS Linux image, the AWS CLI is pre-installed, if not install it using the below link.

* [Install or update the latest version of the AWS CLI - AWS Command Line Interface (amazon.com)](https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html)

1. So, now we need to configure the AWS credentials, which were used to deploy the cluster initially using the following commands.

* aws configure

Then, enter the respective access & secret keys, if required also the default region and verify the user identity using the below command.

* aws sts get-caller-identity

1. Then, check if the cluster state is active and configure the kubectl file as follows.

* aws eks --region <region\_name> describe-cluster --name <cluster-name> --query cluster.status
* aws eks --region <region\_name> update-kubeconfig --name <cluster-name>

1. Now, install Kubectl in your Linux EC2/ Cloud Shell terminal using the following instructions.

* curl -O <https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.1/2023-09-14/bin/linux/amd64/kubectl>
* chmod +x ./kubectl
* mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH

1. After successful installation of Kubectl, we now need to setup the eksctl in our instance using the below commands.

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

1. Now, once the eksctl & kubectl setup is done, we need to configure our pods as follows.

* kubectl get nodes
* kubectl get svc
* kubectl get nodes –watch

1. Then, make use of yaml files or any suitable application deployment setup files to deploy the service. Here, we make use of 2 yaml files, one file to create a service and the second to deploy the service.
2. Create a yaml file with following configuration to create a service.

* vi 2048-pod.yaml

Ex:

apiVersion: v1

kind: Pod

metadata:

  name: 2048-pod

  labels:

    app: 2048-ws

spec:

  containers:

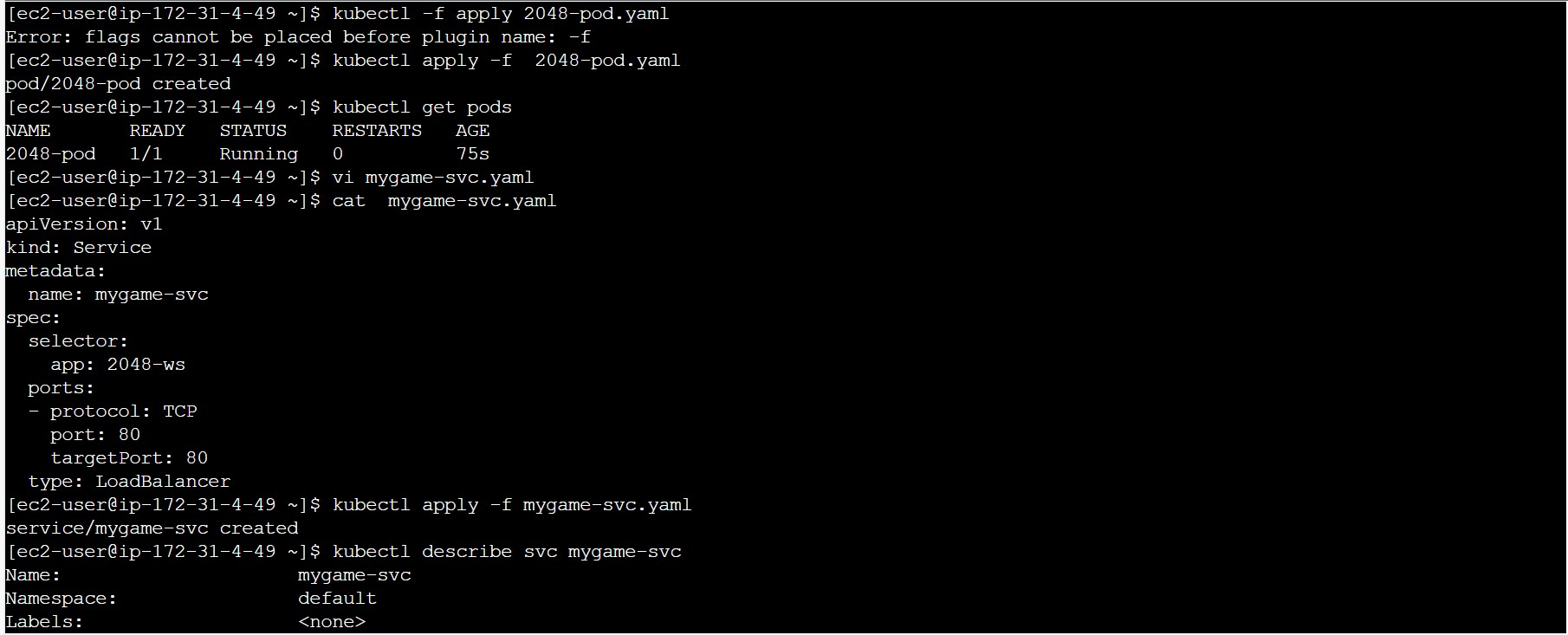
  - name: 2048-container

    image: blackicebird/2048

    ports:

      - containerPort: 80

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



1. Create another yaml file, which is used to deploy the service.

* vi mygame-svc.yaml

Ex:

apiVersion: v1

kind: Service

metadata:

  name: mygame-svc

spec:

  selector:

    app: 2048-ws

  ports:

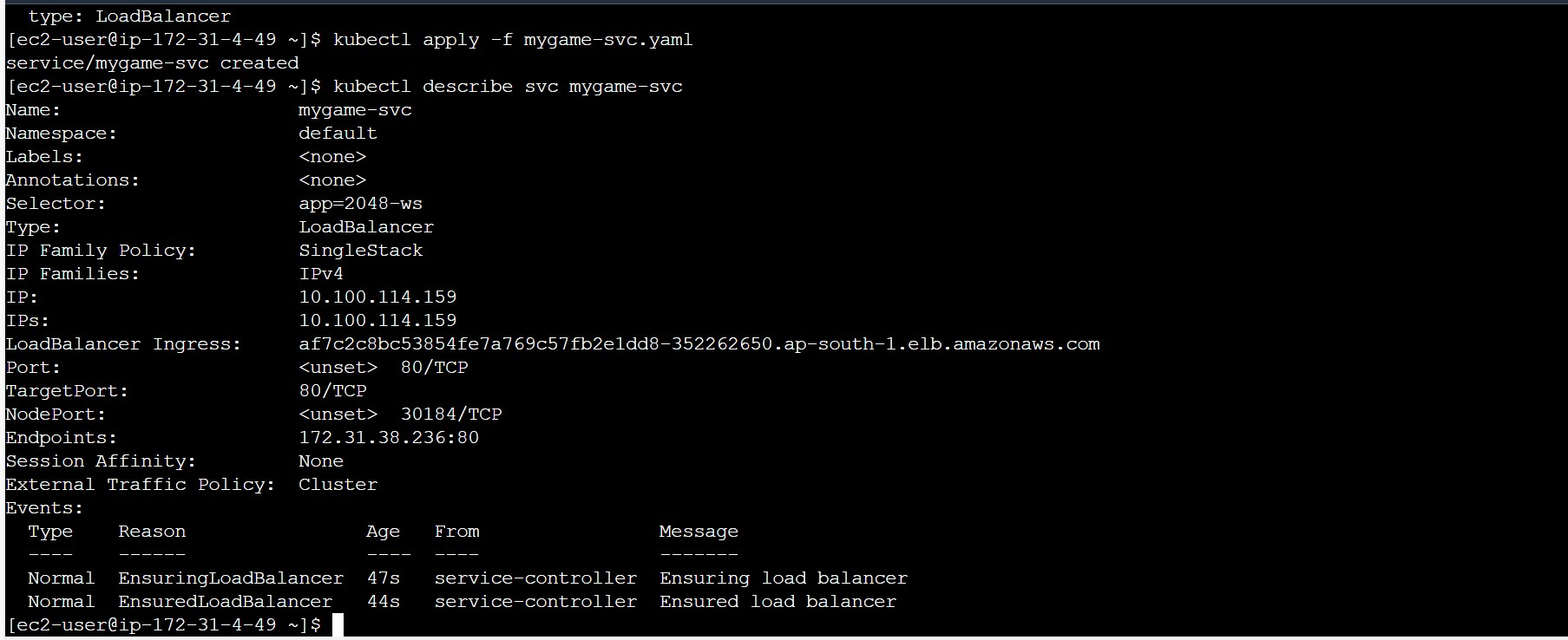
  - protocol: TCP

    port: 80

    targetPort: 80

  type: LoadBalancer

* cat mygame-svc.yaml
* kubectl apply -f mygame-svc.yaml
* kubectl describe svc mygame-svc



1. Now, copy the link opposite to Load Balancer Ingress in the CLI, or move to Load Balancer in EC2 section of AWS console and copy the DNS Name of the LB and , and then render the service in the web to deploy the application. Make sure that the Port number 80, 8080 are enabled in the Cluster’s Security Group.

