**Creating an EKS cluster: (things we need to configure)**

1. Cluster name and K8s version
2. IAM role for the cluster (so it has necessary privileges to perform operations like provisioning nodes, accessing storage and secrets)
3. Select VPC and subnets
4. Define security group for cluster. (To control the traffic going to and from the cluster.)

After creating the EKS cluster, we need to create the worker nodes.

**Creating worker nodes**:

1. Create node group (group of nodes that we’re going to use as worker nodes within our K8s environment)
2. Select Instance type for ec2 instances
3. Define min/max number of nodes in our node group
4. Specify EKS cluster to connect to

**Connecting to cluster:**

After we configured the control plane nodes and worker nodes, the next thing we need to do is to connect to the EKS cluster from our local machine. (So we can run kubectl commands on our newly created cluster). Go to console and get the connection information or the cluster. Run the below command with the information we have.

**Kubectl config set-cluster**

**Methods to create EKS cluster:**

1. AWS console (long and tedious process)
2. eksctl - CLI for Amazon EKS (Quick and Efficient)

**eksctl create cluster** – one single command to setup the cluster its going to provision our VPC’s, subnets, the control plane, the worker nodes and provisioning the node groups.

eksctl create cluster --name cluster1 --nodegroup-name ng1 --region ap-south-2 --node-type t3.small

eksctl delete cluster < cluster name > – to delete the cluster.

1. IaC – Infrastructure as code tools – Terraform

**AWS Concepts:**

VPC – Virtual private network

We can create a subnet within the VPC, the subnet will be allocated to the EC2 machine.

To assign an IP address, the elastic IP address is allocated to the EC2. To associate a public IP address, the VPC should be attached with an internet gateway.

An Internet gateway is necessary to establish connectivity between the VPC & the internet.

We need to ensure that the route table associated with the VPC, has a route to the internet gateway.

Add a route with a destination of 0.0.0.0/0 and select the internet gateway as the target.

Create an IAM role with policies (EC2, ADMINISTRATOR, CLOUDFORMATION, IAM) full access policies and add the role to the ec2.

**Elastic IP:** Static public IPv4 address that we can allocate to our resources such as EC2.

**Ingress:** Ingress exposes the http and https routes from outside the cluster to the services within the cluster. Traffic routing is controlled by rules defined in the ingress resource.

**Ingress controller:** In order for the ingress to work we need ingress controller.

Used helm:

Install helm:

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

[ec2-user@ip-172-31-0-113 ~]$ chmod +x get\_helm.sh

[ec2-user@ip-172-31-0-113 ~] $ ./get\_helm.sh

grep: warning: stray \ before "

Downloading https://get.helm.sh/helm-v3.12.2-linux-amd64.tar.gz

Verifying checksum... Done.

Preparing to install helm into /usr/local/bin

helm installed into /usr/local/bin/helm

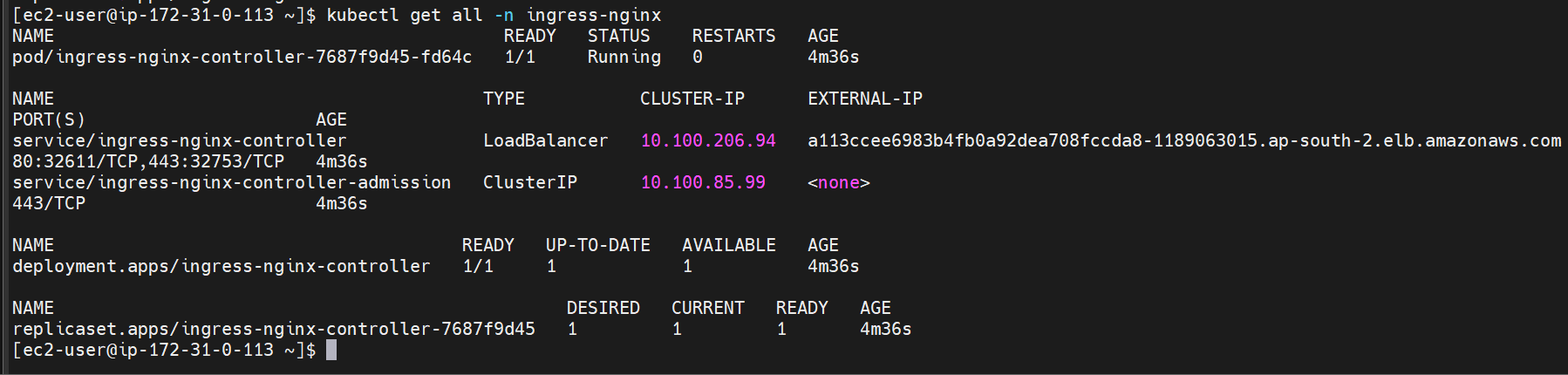
**Installing Ingress-nginx-controller**

helm upgrade --install ingress-nginx ingress-nginx \

--repo <https://kubernetes.github.io/ingress-nginx> \

--namespace ingress-nginx --create-namespace

kubectl get all -n ingress-nginx



helm list -n ingress-nginx to see the helm release.

Deployed ingress rule and a sample application: Below are the yaml:



Tls certificates generated should be specified in the ingress rule as secret.

**To generate a self-signed certificate:**

# Generate the private key

openssl genpkey -algorithm RSA -out selfsigned.key

# Generate the CSR using the private key

openssl req -new -key selfsigned.key -subj "/CN=ap-south-2.elb.amazonaws.com" -out selfsigned.csr (CN is the domain name)

# Create the self-signed certificate (validity: 365 days)

openssl x509 -req -days 365 -in selfsigned.csr -signkey selfsigned.key -out selfsigned.crt

To generate secret:

Kubectl create secret tls <secret-name> --cert <.cert file> --key <.key file>

Ex: kubectl create secret tls self-signed --cert tls.crt --key tls.key

To clean up the resources, uninstall the release.

helm uninstall ingress-nginx -n ingress-nginx.