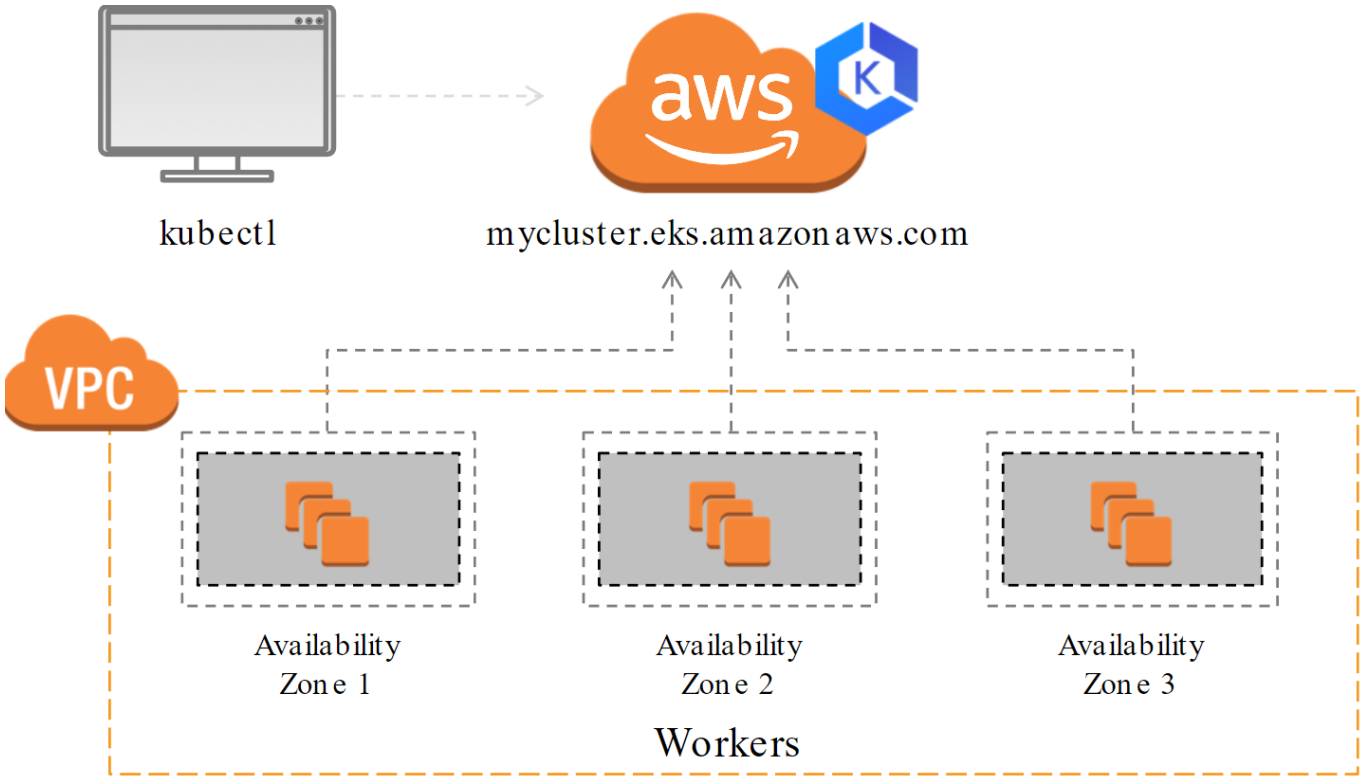
Table of Contents

What is EKS

- AWS EKS is a `Managed Kubernetes Service` from Amazon, which means AWS manages the Master Nodes.
- All the necessary applications/services are already pre-installed like the container runtime or master processes and in addition it also takes care of scaling and backups.
- You only create the Worker Nodes which are EC2 instances.

--

EKS ARCHITECTURE FOR CONTROL PLANE AND WORKER NODE COMMUNICATION



--

Steps to create EKS

- To create a K8s cluster in EKS you need to do following steps:
- Create a VPC
- Create an IAM role with Security Group (or in other words: create AWS user with list of permissions)
- Create Cluster Control Plane - Master Nodes
    - choose basic information like cluster name and k8s version
    - choose region and VPC for your cluster
    - set security
- Create Worker Nodes and connect to cluster
    - The Worker Nodes are some EC2 instances with CPU and storage resources.
    - Create as a Node Group
    - Choose cluster it will attach to
    - Define Security Group, select instance type etc.

> With `NodeGroup` you have autoscaling, depending on how much load the cluster has new Worker Nodes will automatically added or removed in the cluster.

--

EKSCTL CLI

- Execute below commands in either one of the below options:

    - AWS CloudShell (Uses AWS Console Login IAM Credentials)
    - Local Linux Machine (Uses IAM User Credentials configured)
    - Amazon EC2 Linux Machine (Uses AWS IAM Role Permissions)

- Download the `eksctl` on your machine linux cli

```
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
```

- The IAM account used for EKS cluster creation should have these minimal access levels.

AWS Service	Access Level
CloudFormation	Full Access
EC2	**Full:** Tagging **Limited:** List, Read, Write
EC2 Auto Scaling	**Limited:** List, Write
EKS	Full Access
IAM	**Limited:** List, Read, Write, Permissions Management
Systems Manager	**Limited:** List, Read

--

- Command to create EKS Cluster

```
eksctl create cluster \
  --name test-eks-cluster \
  --version 1.25 \
  --region ap-south-1 \
  --nodegroup-name eks-worker-nodegroup \
  --node-type t4g.small \
  --nodes 2

#####OUTPUT#######
2022-01-26 01:53:53 [ℹ]  eksctl version 0.80.0
2022-01-26 01:53:53 [ℹ]  using region ap-south-1
2022-01-26 01:53:54 [ℹ]  setting availability zones to [ap-south-1b ap-south-1a
ap-south-1c]
2022-01-26 01:53:54 [ℹ]  subnets for ap-south-1b - public:192.168.0.0/19
private:192.168.96.0/19
2022-01-26 01:53:54 [ℹ]  subnets for ap-south-1a - public:192.168.32.0/19
private:192.168.128.0/19
2022-01-26 01:53:54 [ℹ]  subnets for ap-south-1c - public:192.168.64.0/19
private:192.168.160.0/19
2022-01-26 01:53:54 [ℹ]  nodegroup "ng-2564739d" will use "" [AmazonLinux2/1.21]
2022-01-26 01:53:54 [ℹ]  using Kubernetes version 1.21
2022-01-26 01:53:54 [ℹ]  creating EKS cluster "test-eks-cluster" in "ap-south-1"
region with managed nodes
2022-01-26 01:53:54 [ℹ]  will create 2 separate CloudFormation stacks for cluster
```

```
itself and the initial managed nodegroup
2022-01-26 01:53:54 [ℹ]  if you encounter any issues, check CloudFormation console
or try 'eksctl utils describe-stacks --region=ap-south-1 --cluster=test-eks-
cluster'
2022-01-26 01:53:54 [ℹ]  Kubernetes API endpoint access will use default of
{publicAccess=true, privateAccess=false} for cluster "test-eks-cluster" in "ap-
south-1"
2022-01-26 01:53:54 [ℹ]  CloudWatch logging will not be enabled for cluster "test-
eks-cluster" in "ap-south-1"
2022-01-26 01:53:54 [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=test-eks-cluster'
2022-01-26 01:53:54 [ℹ]
2 sequential tasks: { create cluster control plane "test-eks-cluster",
    2 sequential sub-tasks: {
        wait for control plane to become ready,
        create managed nodegroup "ng-2564739d",
    }
}
2022-01-26 01:53:54 [ℹ]  building cluster stack "eksctl-test-eks-cluster-cluster"
2022-01-26 01:53:55 [ℹ]  deploying stack "eksctl-test-eks-cluster-cluster"
2022-01-26 01:54:25 [ℹ]  waiting for CloudFormation stack "eksctl-test-eks-
cluster-cluster"

2022-10-19 02:37:29 [ℹ]  waiting for the control plane to become ready
2022-10-19 02:37:30 [✔]  saved kubeconfig as "/home/cloudshell-
user/.kube/config"
2022-10-19 02:37:30 [ℹ]  no tasks
2022-10-19 02:37:30 [✔]  all EKS cluster resources for "test-eks-cluster" have
been created
2022-10-19 02:37:31 [ℹ]  nodegroup "eks-worker-nodegroup" has 2 node(s)
2022-10-19 02:37:31 [ℹ]  node "ip-192-168-31-69.ap-south-1.compute.internal" is
ready
2022-10-19 02:37:31 [ℹ]  node "ip-192-168-62-91.ap-south-1.compute.internal" is
ready
2022-10-19 02:37:31 [ℹ]  waiting for at least 2 node(s) to become ready in "eks-
worker-nodegroup"
2022-10-19 02:37:31 [ℹ]  nodegroup "eks-worker-nodegroup" has 2 node(s)
2022-10-19 02:37:31 [ℹ]  node "ip-192-168-31-69.ap-south-1.compute.internal" is
ready
2022-10-19 02:37:31 [ℹ]  node "ip-192-168-62-91.ap-south-1.compute.internal" is
ready
2022-10-19 02:37:34 [ℹ]  kubectl command should work with "/home/cloudshell-
user/.kube/config", try 'kubectl get nodes'
2022-10-19 02:37:34 [✔]  EKS cluster "test-eks-cluster" in "ap-south-1" region
is ready

#####OUTPUT#######
```

--

- Install `kubectl` linux utility on your local linux client.

```
curl -o kubectl https://s3.us-west-2.amazonaws.com/amazon-eks/1.21.5/2022-01-
21/bin/linux/amd64/kubectl
chmod +x ./kubectl
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export
PATH=$PATH:$HOME/bin
echo 'export PATH=$PATH:$HOME/bin' >> ~/.bashrc
```

```
# View the Local kubeconfig file
kubectl config view
# This command shows content inside the ~/.kube/config file
#####OUTPUT#######
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: DATA+OMITTED
    server: https://00D6914D1CB90D39116178B3A9A2ECA6.yl4.ap-south-
1.eks.amazonaws.com
  name: test-eks-cluster.ap-south-1.eksctl.io
contexts:
- context:
    cluster: test-eks-cluster.ap-south-1.eksctl.io
    user: ansible-user@test-eks-cluster.ap-south-1.eksctl.io
  name: ansible-user@test-eks-cluster.ap-south-1.eksctl.io
current-context: ansible-user@test-eks-cluster.ap-south-1.eksctl.io
kind: Config
preferences: {}
users:
- name: ansible-user@test-eks-cluster.ap-south-1.eksctl.io
  user:
    exec:
      apiVersion: client.authentication.k8s.io/v1alpha1
      args:
      - eks
      - get-token
      - --cluster-name
      - test-eks-cluster
      - --region
      - ap-south-1
      command: aws
      env:
      - name: AWS_STS_REGIONAL_ENDPOINTS
        value: regional
      provideClusterInfo: false
#####OUTPUT#######
```

--

> When you create an Amazon EKS cluster, the AWS Identity and Access Management (IAM) entity user or role that creates the cluster, is automatically granted `system:masters` permissions in the cluster's

> `role-based access control` (RBAC) configuration in the Amazon EKS control plane.This IAM entity doesn't appear in any visible configuration, so make sure to keep track of which IAM entity originally created the cluster.

- Validate the CloudFormation Template Resources created with above command.
  - Stack Name : **eksctl-test-eks-cluster-cluster**
  - Stack Name : **eksctl-test-eks-cluster-nodegroup-eks-worker-nodegroup**

--

Verify Cluster, NodeGroup in EKS Management Console

- Go to Services -> **Elastic Kubernetes Service**

List Worker Nodes

```
# List EKS clusters
eksctl get cluster

# List NodeGroups in a cluster
eksctl get nodegroup --cluster=<clusterName>

# List Nodes in current kubernetes cluster
kubectl get nodes
[cloudshell-user@ip-10-0-173-52 ~]$ kubectl get nodes
Kubeconfig user entry is using deprecated API version
client.authentication.k8s.io/v1alpha1. Run 'aws eks update-kubeconfig' to update.
NAME                                           STATUS   ROLES    AGE     VERSION
ip-192-168-4-237.ap-south-1.compute.internal   Ready    <none>   7m58s   v1.21.12-
eks-5308cf7
ip-192-168-85-16.ap-south-1.compute.internal   Ready    <none>   8m5s    v1.21.12-
eks-5308cf7

# Our kubectl context should be automatically changed to new cluster
kubectl config view --minify
```

--

kubectl commands

```
kubectl get pods -n kube-system
kubectl get pods -n kube-system -o wide
```

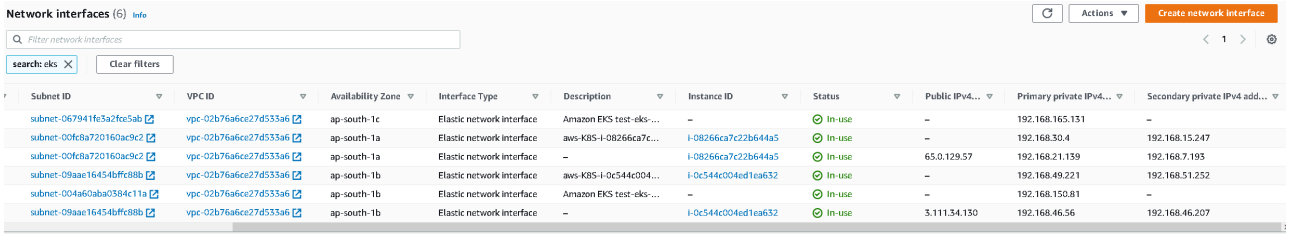
- Create the Deployment by running the following commands:
  - Reference : run-stateless-application-deployment

```
```

```
kubectl apply -f https://k8s.io/examples/controllers/nginx-deployment.yaml
wget https://k8s.io/examples/controllers/nginx-deployment.yaml

[cloudshell-user@ip-10-0-50-112 ~]$ kubectl get events
LAST SEEN   TYPE      REASON              OBJECT
MESSAGE
2m48s       Normal    Scheduled           pod/nginx-deployment-66b6c48dd5-qkst9
Successfully assigned default/nginx-deployment-66b6c48dd5-qkst9 to ip-192-168-21-
139.ap-south-1.compute.internal
2m47s       Normal    Pulling             pod/nginx-deployment-66b6c48dd5-qkst9
Pulling image "nginx:1.14.2"
2m38s       Normal    Pulled              pod/nginx-deployment-66b6c48dd5-qkst9
Successfully pulled image "nginx:1.14.2" in 9.12123707s
2m37s       Normal    Created             pod/nginx-deployment-66b6c48dd5-qkst9
Created container nginx
2m37s       Normal    Started             pod/nginx-deployment-66b6c48dd5-qkst9
Started container nginx
37s         Warning   FailedScheduling    pod/nginx-deployment-66b6c48dd5-x92pj
0/2 nodes are available: 2 Too many pods.
2m48s       Normal    Scheduled           pod/nginx-deployment-66b6c48dd5-zb9bb
Successfully assigned default/nginx-deployment-66b6c48dd5-zb9bb to ip-192-168-46-
56.ap-south-1.compute.internal
2m47s       Normal    Pulling             pod/nginx-deployment-66b6c48dd5-zb9bb
Pulling image "nginx:1.14.2"
2m37s       Normal    Pulled              pod/nginx-deployment-66b6c48dd5-zb9bb
Successfully pulled image "nginx:1.14.2" in 9.470360458s
2m37s       Normal    Created             pod/nginx-deployment-66b6c48dd5-zb9bb
Created container nginx
2m37s       Normal    Started             pod/nginx-deployment-66b6c48dd5-zb9bb
Started container nginx
2m48s       Normal    SuccessfulCreate    replicaset/nginx-deployment-66b6c48dd5
Created pod: nginx-deployment-66b6c48dd5-zb9bb
2m48s       Normal    SuccessfulCreate    replicaset/nginx-deployment-66b6c48dd5
Created pod: nginx-deployment-66b6c48dd5-qkst9
2m48s       Normal    SuccessfulCreate    replicaset/nginx-deployment-66b6c48dd5
Created pod: nginx-deployment-66b6c48dd5-x92pj
2m48s       Normal    ScalingReplicaSet   deployment/nginx-deployment
Scaled up replica set nginx-deployment-66b6c48dd5 to 3

kubectl describe pods -n default > pods.txt
cat pods.txt

Name:         nginx-deployment-66b6c48dd5-qkst9
Namespace:    default
Priority:     0
Node:         ip-192-168-21-139.ap-south-1.compute.internal/192.168.21.139
Start Time:   Wed, 02 Feb 2022 19:29:23 +0000
Labels:       app=nginx
              pod-template-hash=66b6c48dd5
Annotations:  kubernetes.io/psp: eks.privileged
Status:       Running
IP:           192.168.15.247
IPs:
```

```
  IP:           192.168.15.247
Controlled By:  ReplicaSet/nginx-deployment-66b6c48dd5

Name:           nginx-deployment-66b6c48dd5-x92pj
Namespace:      default
Priority:       0
Node:           <none>
Labels:         app=nginx
                pod-template-hash=66b6c48dd5
Annotations:    kubernetes.io/psp: eks.privileged
Status:         Pending
IP:
IPs:            <none>
Controlled By:  ReplicaSet/nginx-deployment-66b6c48dd5

Name:         nginx-deployment-66b6c48dd5-zb9bb
Namespace:    default
Priority:     0
Node:         ip-192-168-46-56.ap-south-1.compute.internal/192.168.46.56
Start Time:   Wed, 02 Feb 2022 19:29:23 +0000
Labels:       app=nginx
              pod-template-hash=66b6c48dd5
Annotations:  kubernetes.io/psp: eks.privileged
Status:       Running
IP:           192.168.51.252
IPs:
  IP:           192.168.51.252
Controlled By:  ReplicaSet/nginx-deployment-66b6c48dd5

kubectl get pods -o wide
```

--

- The above command shows pods details where the pod gets an IP from ENI in the Subnet.
- 

--

- AWS EKS supports native VPC networking with the Amazon VPC Container Network Interface (CNI) plugin for Kubernetes.
- Using this plugin allows Kubernetes Pods to have the same IP address inside the pod as they do on the VPC network.
- For more information, see amazon-vpc-cni-k8s and Proposal: CNI plugin for Kubernetes networking over AWS VPC on GitHub.

- The Amazon VPC CNI plugin is fully supported for use on Amazon EKS and self-managed Kubernetes clusters on AWS.
- Refer the ENI Limit as per Instance Type : https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-eni.html#AvailableIpPerENI
    - The formula for defining the maximum number of Pods per EC2 Node instance is as follows:
        - **N * (M-1) + 2**
            - **N** is the number of Elastic Network Interfaces (ENI) of the instance type
            - **M** is the number of IP addresses per ENI.
        - For e.g. **t2.micro** instance, this calculation is 2 * (2-1) + 2 = 4 Pods

--

- Use below command to get similar details:

```
aws ec2 describe-instance-types --filters "Name=instance-type,Values=t4g.*" --
query "InstanceTypes[].{Type: InstanceType, MaxENI:
NetworkInfo.MaximumNetworkInterfaces, IPv4addr:
NetworkInfo.Ipv4AddressesPerInterface}" --output table
-------------------------------------
|        DescribeInstanceTypes        |
+----------+----------+---------------+
| IPv4addr | MaxENI   |     Type      |
+----------+----------+---------------+
|  2       |  2       |  t4g.nano     |
|  6       |  3       |  t4g.medium   |
|  12      |  3       |  t4g.large    |
|  15      |  4       |  t4g.2xlarge  |
|  4       |  3       |  t4g.small    |
|  2       |  2       |  t4g.micro    |
|  15      |  4       |  t4g.xlarge   |
+----------+----------+---------------+
```

```
kubectl get pods --all-namespaces -o wide
kubectl get pods --all-namespaces -o wide | grep -i running
[cloudshell-user@ip-10-0-50-112 ~]$ kubectl get pods --all-namespaces -o wide |
grep -i running
default       nginx-deployment-66b6c48dd5-qkst9  1/1     Running   0  38m
192.168.15.247 ip-192-168-21-139.ap-south-1.compute.internal
default       nginx-deployment-66b6c48dd5-zb9bb  1/1     Running   0  38m
192.168.51.252 ip-192-168-46-56.ap-south-1.compute.internal
kube-system   aws-node-jls58                     1/1     Running   0  125m
192.168.46.56  ip-192-168-46-56.ap-south-1.compute.internal
kube-system   aws-node-tmwl2                     1/1     Running   0  125m
192.168.21.139 ip-192-168-21-139.ap-south-1.compute.internal
kube-system   coredns-7f95bc96cc-dl6dn           1/1     Running   0  135m
192.168.46.207 ip-192-168-46-56.ap-south-1.compute.internal
kube-system   coredns-7f95bc96cc-fj56b           1/1     Running   0  135m
192.168.7.193  ip-192-168-21-139.ap-south-1.compute.internal
```

```
kube-system   kube-proxy-dqlph                    1/1     Running   0  125m
192.168.21.139 ip-192-168-21-139.ap-south-1.compute.internal
kube-system   kube-proxy-xlpfk                    1/1     Running   0  125m
192.168.46.56  ip-192-168-46-56.ap-south-1.compute.internal
[cloudshell-user@ip-10-0-50-112 ~]$
```

--

Kubernetes Application Deployment

Docker Image Creation and Push to Registry

- Below Steps can be performed on any Docker Host or CI Job.
- Clone Repo: https://github.com/devops-practicals/notes-app-cicd.git

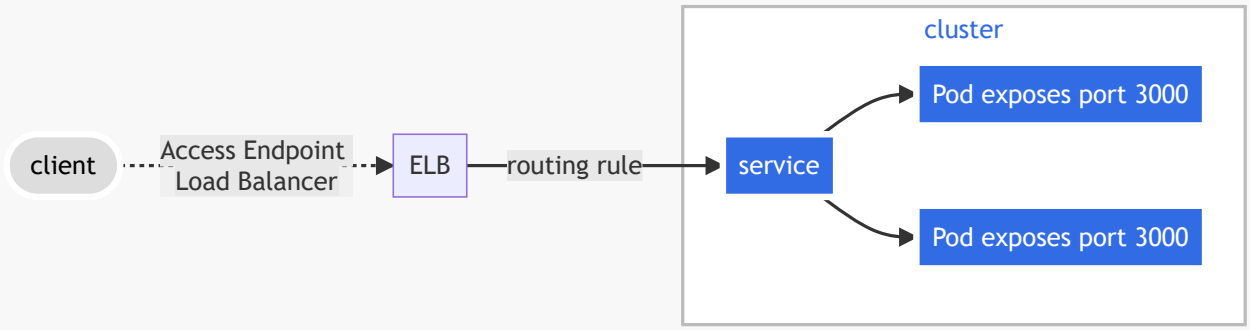
```
cd notes-app-cicd

sudo docker login
# Enter DockerHub Username and Password

# Build Docker Image

sudo docker build -t notes-app .

# Tag Docker Image
sudo docker tag notes-app:latest <Dockerhub_Username>/notes-app:latest

# Push the Docker image
sudo docker push <Dockerhub_Username>/notes-app:latest
```

- Once Image is available in the Image Registry i.e DockerHub/ECR, specify this image Name:Tag in the Kubernetes Deployment YAML Manifest Files.

```
# Validate the image in deployment.yaml file
apiVersion: apps/v1
kind: Deployment
metadata:
  name: notes-app-deployment
  labels:
    app: notes-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: notes-app
  template:
    metadata:
      labels:
        app: notes-app
```

```
    spec:
      containers:
      - name: notes-app-deployment
        image: <Dockerhub_Username>/notes-app:latest
        resources:
          requests:
            cpu: "100m"
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 3000

# Validate the service.yaml file
apiVersion: v1
# Indicates this as a service
kind: Service
metadata:
 # Service name
 name: notes-app-deployment
spec:
 selector:
   # Selector for Pods
   app: notes-app
 ports:
   # Port Map
 - port: 80
   targetPort: 3000
   protocol: TCP
 type: LoadBalancer

# Apply the manifest files with the following commands. Starting with deployment
and then service yaml file.
sudo kubectl apply -f deployment.yaml

sudo kubectl apply -f service.yaml

sudo kubectl get pods
#
NAME                                    READY   STATUS    RESTARTS   AGE
notes-app-deployment-568df5649-5vdjc    1/1     Running   0          16s
notes-app-deployment-568df5649-g286v    1/1     Running   0          16s

sudo kubectl get svc
NAME                  TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)
AGE
kubernetes            ClusterIP      10.96.0.1       <none>        443/TCP
14m
notes-app-deployment   LoadBalancer   10.111.50.144   <pending>     80:31263/TCP
5s

# Access the application UI on Host Public IP with port as 31263
# http://3.110.78.93:31263/

# If you executed this Kubernetes deployment and service on EKS Cluster, a ELB
Endpoint will be created.
```

```
# Access the application on the ELB Endpoint.
[cloudshell-user@ip-10-2-86-100 ~]$ kubectl describe svc notes-app-deployment
Name:                     notes-app-deployment
Namespace:                default
Labels:                   <none>
Annotations:              <none>
Selector:                 app=notes-app
Type:                     LoadBalancer
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.100.113.118
IPs:                      10.100.113.118
LoadBalancer Ingress:     a236feab6f55f47d99c3fb72709686f9-1489323979.ap-south-
1.elb.amazonaws.com
Port:                     <unset>  80/TCP
TargetPort:               3000/TCP
NodePort:                 <unset>  31263/TCP
Endpoints:                192.168.28.162:3000,192.168.43.57:3000
Session Affinity:         None
External Traffic Policy:  Cluster
Events:
  Type    Reason                Age   From                Message
  ----    ------                ----  ----                -------
  Normal  EnsuringLoadBalancer  43m   service-controller  Ensuring load balancer
  Normal  EnsuredLoadBalancer   43m   service-controller  Ensured load balancer

# If you delete the kubernetes service object, the ELB endpoint is deleted.
kubectl delete svc notes-app-deployment
```



> Once Testing of EKS Cluster Creation is done, make sure to Delete the cluster when not in use.

```
eksctl delete cluster --name test-eks-cluster
```

EKS Cluster Pricing

- EKS is not free (Unlike other AWS Services), In short, no free-tier for EKS.

EKS Cluster Pricing

- Below is the EKS Cluster Pricing:
  - $0.10/hour
  - $2.4/day
  - $72/month

EKS Worker Nodes Pricing-EC2

- You pay for AWS resources (e.g. EC2 instances or EBS volumes)
- For **t3.medium** in N.Virginia
  - $0.0416 per Hour
  - Per Day: $0.9984 - Approximately $1
  - Per Month: $30 per 1 t3.medium Node.
- Reference: EC2 On-Demand Pricing

--

EKS Case Study:

- meity-gov-india-case-study
- Architectural lessons from CoWIN platform
- cowin-public-v2

--

Reference

- EKS Cost Optimization
- Kubernetes instance calculator
- Resource Limits and Requests

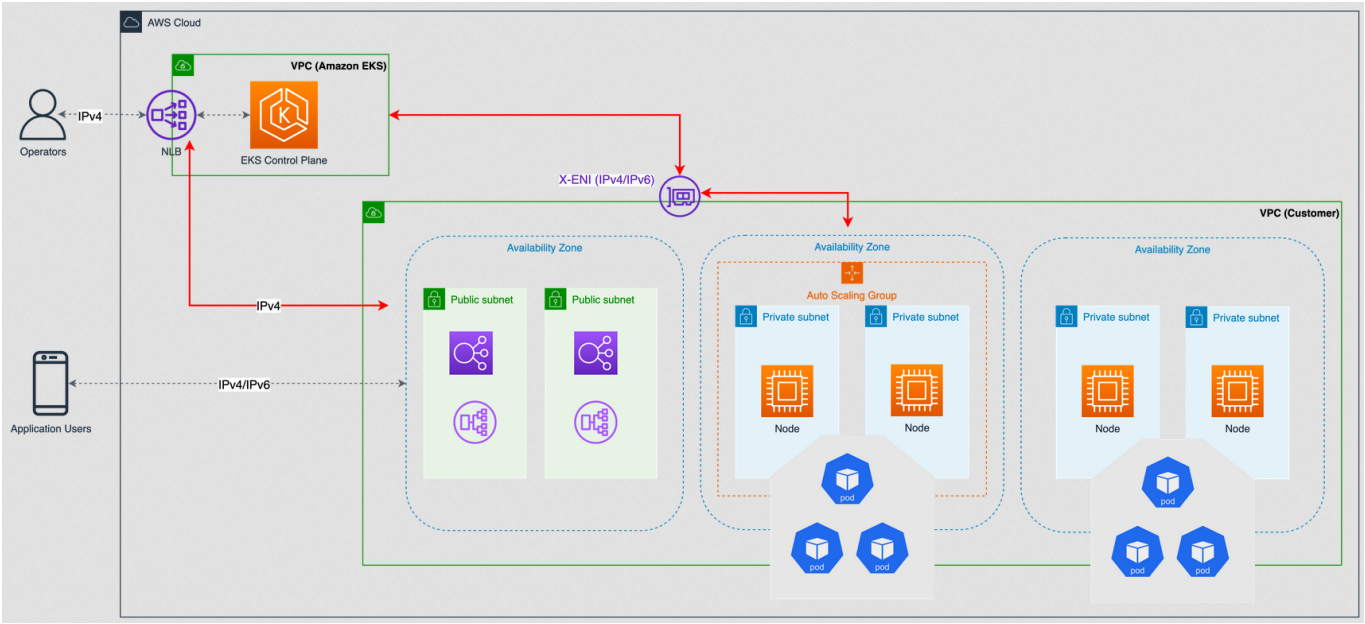
--

Reference:

- AWS Graviton2 t4g.small free for all AWS Accounts until December 31, 2023
  - Until December 31, 2023, all AWS customers will be enrolled automatically in the t4g.small Free Trial as detailed in the AWS Free Tier.
  - During the free-trial period, customers who run a t4g.small instance will automatically get 750 free hours per month deducted from their bill during each month.

--

- aws-eks-best-practices



- eks-cluster-connection