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 Cluster Creation using Terraform

- For creating the EKS cluster using Terraform, Check for Latest Kubernetes Version here
- Follow steps mentioned in this Document: https://developer.hashicorp.com/terraform/tutorials/kubernetes/eks
- Modify the **eks-cluster.tf** to include **instance_types = ["t4g.small"]**
- Modify the **eks-cluster.tf** to only have one **NodeGroup**
- Execute:
  - terraform init
  - terraform plan
  - terraform apply
- Validate all resources created in AWS Region **us-east-2**.
- Make sure **teraform destroy** is executed to delete all infrastructure.
- https://github.com/AdminTurnedDevOps/Kubernetes-Quickstart-Environments/tree/main/aws
- 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 --region
region-code --name my-cluster' 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
```

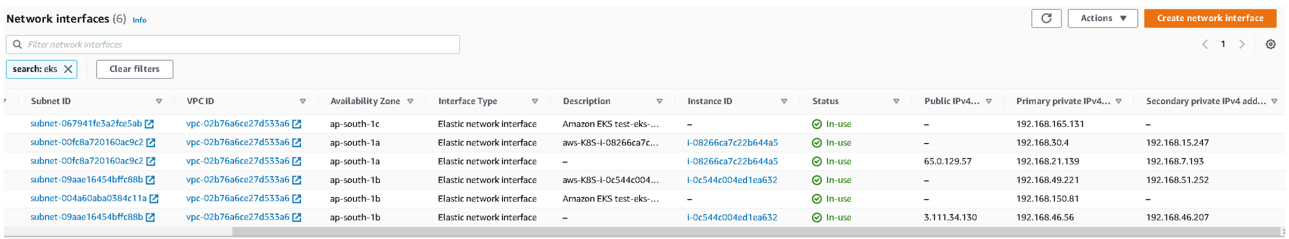
```
MESSAGE
2m48s       Normal    Scheduled           pod/nginx-deployment-66b6c48dd5-qkst9
Successfully assigned default/nginx-deployment-66b6c48dd5-qkst9 to ip-192-168-21-
139.ap-south-1.compute.internal
2m47s       Normal    Pulling             pod/nginx-deployment-66b6c48dd5-qkst9
Pulling image "nginx:1.14.2"
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
Containers:
  nginx:
    Container ID:
docker://cb8aaf025035aa3a81d05f1598efbfc945bbe4ae01558a0ee0dee02942f32175
```

```
    Image:          nginx:1.14.2
    Image ID:       docker-
pullable://nginx@sha256:f7988fb6c02e0ce69257d9bd9cf37ae20a60f1df7563c3a2a6abe24160
306b8d
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Wed, 02 Feb 2022 19:29:34 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-5qrpp
(ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  kube-api-access-5qrpp:
    Type:                    Projected (a volume that contains injected data from
multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for
300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists
for 300s
Events:
  Type    Reason     Age    From               Message
  ----    ------     ----   ----               -------
  Normal  Scheduled  9m3s   default-scheduler  Successfully assigned
default/nginx-deployment-66b6c48dd5-qkst9 to ip-192-168-21-139.ap-south-
1.compute.internal
  Normal  Pulling    9m2s   kubelet            Pulling image "nginx:1.14.2"
  Normal  Pulled     8m53s  kubelet            Successfully pulled image
"nginx:1.14.2" in 9.12123707s
  Normal  Created    8m52s  kubelet            Created container nginx
  Normal  Started    8m52s  kubelet            Started container nginx

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
Containers:
  nginx:
    Image:        nginx:1.14.2
    Port:         80/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-x7mxn
(ro)
Conditions:
  Type           Status
  PodScheduled   False
Volumes:
  kube-api-access-x7mxn:
    Type:                    Projected (a volume that contains injected data from
multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for
300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists
for 300s
Events:
  Type     Reason            Age                  From               Message
  ----     ------            ----                 ----               -------
  Warning  FailedScheduling  52s (x9 over 9m3s)  default-scheduler  0/2 nodes are
available: 2 Too many pods.

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
Containers:
  nginx:
    Container ID:
docker://9558a069050aba846aa62160509592ebb85dd82d1d90504bc5b03665a0259d47
    Image:          nginx:1.14.2
    Image ID:       docker-
```

```
pullable://nginx@sha256:f7988fb6c02e0ce69257d9bd9cf37ae20a60f1df7563c3a2a6abe24160
306b8d
    Port:          80/TCP
    Host Port:     0/TCP
    State:         Running
      Started:     Wed, 02 Feb 2022 19:29:34 +0000
    Ready:         True
    Restart Count:  0
    Environment:   <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-tnsxj
(ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  kube-api-access-tnsxj:
    Type:                    Projected (a volume that contains injected data from
multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for
300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists
for 300s
Events:
  Type    Reason     Age    From               Message
  ----    ------     ----   ----               -------
  Normal  Scheduled  9m3s   default-scheduler  Successfully assigned
default/nginx-deployment-66b6c48dd5-zb9bb to ip-192-168-46-56.ap-south-
1.compute.internal
  Normal  Pulling    9m2s   kubelet            Pulling image "nginx:1.14.2"
  Normal  Pulled     8m52s  kubelet            Successfully pulled image
"nginx:1.14.2" in 9.470360458s
  Normal  Created    8m52s  kubelet            Created container nginx
  Normal  Started    8m52s  kubelet            Started container nginx
```

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



Subnet ID	VPC ID	Availability Zone	Interface Type	Description	Instance ID	Status	Public IPv4...	Primary private IPv4...	Secondary private IPv4 add...
subnet-067941fe3a2fce5ab	vpc-02b76a6ce27d533a6	ap-south-1c	Elastic network interface	Amazon EKS test-eks-...	–	In-use	–	192.168.165.131	–
subnet-00fc8a720160ac9c2	vpc-02b76a6ce27d533a6	ap-south-1a	Elastic network interface	aws-K8S-i-08266ca7c...	i-08266ca7c22b644a5	In-use	–	192.168.30.4	192.168.15.247
subnet-00fc8a720160ac9c2	vpc-02b76a6ce27d533a6	ap-south-1a	Elastic network interface	–	i-08266ca7c22b644a5	In-use	65.0.129.57	192.168.21.139	192.168.7.193
subnet-09aae16454bffc88b	vpc-02b76a6ce27d533a6	ap-south-1b	Elastic network interface	aws-K8S-i-0c544c004...	i-0c544c004ed1ea632	In-use	–	192.168.49.221	192.168.51.252
subnet-004a60aba0384c11a	vpc-02b76a6ce27d533a6	ap-south-1b	Elastic network interface	Amazon EKS test-eks-...	–	In-use	–	192.168.150.81	–
subnet-09aae16454bffc88b	vpc-02b76a6ce27d533a6	ap-south-1b	Elastic network interface	–	i-0c544c004ed1ea632	In-use	3.111.34.130	192.168.46.56	192.168.46.207

-

- 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 details on same:

```
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 ~]$
```

- Delete the cluster using

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

Reference

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

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

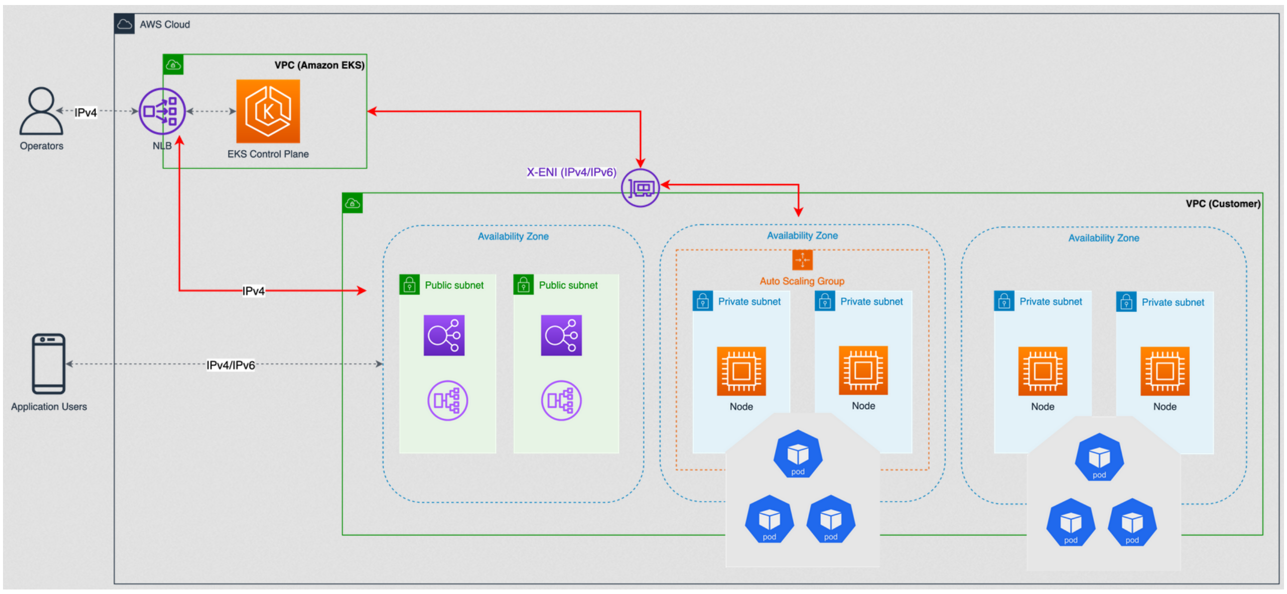
EKS Worker Nodes Pricing-EC2

- You pay for AWS resources (e.g. EC2 instances or EBS volumes)
- T3 Medium Server in N.Virginia
  - $0.0416 per Hour
  - Per Day: $0.9984 - Approximately $1
  - Per Month: $30 per 1 t3.medium server
- Reference: EC2 On-Demand Pricing

EKS Case Study:

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

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
- terraform-eks