

Containerized Nginx Deployment on Amazon Elastic Container Service for Kubernetes (EKS)

This lab demonstrates the steps to deploy and manage a containerized Nginx application on an Amazon Elastic Container Service for Kubernetes cluster.

The procedures are highlighted below:

EKSCTL was installed because it was needed to make API calls to AWS for the provisioning and management of the EKS cluster.

```
elevated shell. When you open the command shell, you should ensure
that you do so with "Run as Administrator" selected. If you are
attempting to use Chocolatey in a non-administrator setting, you
must select a different location other than the default install
location. See
https://docs.chocolatey.org/en-us/choco/setup#non-administrative-install
for details.

Do you want to continue?([Y]es/[N]o): y

Installing the following packages:
eksctl
By installing, you accept licenses for the packages.
eksctl v0.207.0 already installed.
Use --force to reinstall, specify a version to install, or try upgrade.

Chocolatey installed 0/1 packages.
See the log for details (C:\ProgramData\chocolatey\logs\chocolatey.log).

Warnings:
- eksctl - eksctl v0.207.0 already installed.
Use --force to reinstall, specify a version to install, or try upgrade.

Enjoy using Chocolatey? Explore more amazing features to take your
experience to the next level at
https://chocolatey.org/compare
PS C:\Users\gertr> eksctl version
v0.207.0
PS C:\Users\gertr>
```

Kubectl was also installed to interact with the Kubernetes control plane in order to manage applications, nodes and pods in the cluster upon provisioning.

```
PS C:\Users\gertr> kubectl version --client
Client Version: v1.33.0
Kustomize Version: v5.6.0
PS C:\Users\gertr>
```

The EKS cluster was provisioned with the configurations below using the eksctl command line tool

```
Windows PowerShell
PS C:\Users\gertr> aws configure
AWS Access Key ID [*****4EFM]:
AWS Secret Access Key [*****urgy]:
Default region name [us-east-1]:
Default output format [json]:
PS C:\Users\gertr> eksctl create cluster --name my-cluster --region us-east-1 --nodegroup-name my-nodes --node-type t3.medium --nodes-min 1 --nodes-max 4 --managed
2025-05-11 21:09:20 [i] eksctl version 0.207.0
2025-05-11 21:09:20 [i] using region us-east-1
2025-05-11 21:09:21 [i] setting availability zones to [us-east-1c us-east-1f]
2025-05-11 21:09:21 [i] subnets for us-east-1c - public:192.168.0.0/19 private:192.168.64.0/19
2025-05-11 21:09:21 [i] subnets for us-east-1f - public:192.168.32.0/19 private:192.168.96.0/19
2025-05-11 21:09:21 [i] nodegroup "my-nodes" will use "" [AmazonLinux2/1.32]
2025-05-11 21:09:21 [i] using Kubernetes version 1.32
2025-05-11 21:09:21 [i] creating EKS cluster "my-cluster" in "us-east-1" region with managed nodes
2025-05-11 21:09:21 [i] will create 2 separate CloudFormation stacks for cluster itself and the initial managed nodegroup
2025-05-11 21:09:21 [i] if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=us-east-1 --cluster=my-cluster'
2025-05-11 21:09:21 [i] Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "my-cluster" in "us-east-1"
2025-05-11 21:09:21 [i] CloudWatch logging will not be enabled for cluster "my-cluster" in "us-east-1"
2025-05-11 21:09:21 [i] you can enable it with 'eksctl utils update-cluster-logging --enable-types={SPECIFY-YOUR-LOG-TYPES-HERE (e.g. all)} --region=us-east-1 --cluster=my-cluster'
2025-05-11 21:09:21 [i] default addons vpc-cni, kube-proxy, coredns, metrics-server were not specified, will install them as EKS addons
2025-05-11 21:09:21 [i]
2 sequential tasks: { create cluster control plane "my-cluster",
  2 sequential sub-tasks: {
    2 sequential sub-tasks: {
      1 task: { create addons },
      wait for control plane to become ready,
    },
    create managed nodegroup "my-nodes",
  }
}
2025-05-11 21:09:21 [i] building cluster stack "eksctl-my-cluster-cluster"
```

```
Windows PowerShell
2025-05-11 21:14:29 [I] waiting for CloudFormation stack "eksctl-my-cluster-cluster"
2025-05-11 21:15:30 [I] waiting for CloudFormation stack "eksctl-my-cluster-cluster"
2025-05-11 21:16:31 [I] waiting for CloudFormation stack "eksctl-my-cluster-cluster"
2025-05-11 21:17:32 [I] waiting for CloudFormation stack "eksctl-my-cluster-cluster"
2025-05-11 21:17:38 [I] recommended policies were found for "vpc-cni" addon, but since OIDC is disabled on the cluster, eksctl cannot configure the requested permissions; the recommended way to provide IAM permissions for "vpc-cni" addon is via pod identity associations; after addon creation is completed, add all recommended policies to the config file, under 'addon.PodIdentityAssociations', and run 'eksctl update addon'
2025-05-11 21:17:38 [I] creating addon: vpc-cni
2025-05-11 21:17:39 [I] successfully created addon: vpc-cni
2025-05-11 21:17:40 [I] creating addon: kube-proxy
2025-05-11 21:17:41 [I] successfully created addon: kube-proxy
2025-05-11 21:17:42 [I] creating addon: coredns
2025-05-11 21:17:42 [I] successfully created addon: coredns
2025-05-11 21:17:43 [I] creating addon: metrics-server
2025-05-11 21:17:44 [I] successfully created addon: metrics-server
2025-05-11 21:19:50 [I] building managed nodegroup stack "eksctl-my-cluster-nodegroup-my-nodes"
2025-05-11 21:19:52 [I] deploying stack "eksctl-my-cluster-nodegroup-my-nodes"
2025-05-11 21:19:52 [I] waiting for CloudFormation stack "eksctl-my-cluster-nodegroup-my-nodes"
2025-05-11 21:20:23 [I] waiting for CloudFormation stack "eksctl-my-cluster-nodegroup-my-nodes"
2025-05-11 21:21:18 [I] waiting for CloudFormation stack "eksctl-my-cluster-nodegroup-my-nodes"
2025-05-11 21:22:05 [I] waiting for CloudFormation stack "eksctl-my-cluster-nodegroup-my-nodes"
2025-05-11 21:23:26 [I] waiting for CloudFormation stack "eksctl-my-cluster-nodegroup-my-nodes"
2025-05-11 21:23:27 [I] waiting for the control plane to become ready
2025-05-11 21:23:27 [I] saved kubeconfig as "C:\\Users\\gertr\\.kube\\config"
2025-05-11 21:23:27 [I] no tasks
2025-05-11 21:23:27 [I] all EKS cluster resources for "my-cluster" have been created
2025-05-11 21:23:29 [I] nodegroup "my-nodes" has 1 node(s)
2025-05-11 21:23:29 [I] node "ip-192-168-47-54.ec2.internal" is ready
2025-05-11 21:23:29 [I] waiting for at least 1 node(s) to become ready in "my-nodes"
2025-05-11 21:23:29 [I] nodegroup "my-nodes" has 1 node(s)
2025-05-11 21:23:29 [I] node "ip-192-168-47-54.ec2.internal" is ready
2025-05-11 21:23:29 [I] created 1 managed nodegroup(s) in cluster "my-cluster"
2025-05-11 21:23:31 [I] kubectl command should work with "C:\\Users\\gertr\\.kube\\config", try 'kubectl get nodes'
2025-05-11 21:23:31 [I] EKS cluster "my-cluster" in "us-east-1" region is ready
PS C:\\Users\\gertr>
```

A deployment file and a service file was cloned from a git hub repository as shown below

```
Windows PowerShell
2025-05-11 21:21:18 [I] waiting for CloudFormation stack "eksctl-my-cluster-nodegroup-my-nodes"
2025-05-11 21:22:05 [I] waiting for CloudFormation stack "eksctl-my-cluster-nodegroup-my-nodes"
2025-05-11 21:23:26 [I] waiting for CloudFormation stack "eksctl-my-cluster-nodegroup-my-nodes"
2025-05-11 21:23:27 [I] waiting for the control plane to become ready
2025-05-11 21:23:27 [I] saved kubeconfig as "C:\\Users\\gertr\\.kube\\config"
2025-05-11 21:23:27 [I] no tasks
2025-05-11 21:23:27 [I] all EKS cluster resources for "my-cluster" have been created
2025-05-11 21:23:29 [I] nodegroup "my-nodes" has 1 node(s)
2025-05-11 21:23:29 [I] node "ip-192-168-47-54.ec2.internal" is ready
2025-05-11 21:23:29 [I] waiting for at least 1 node(s) to become ready in "my-nodes"
2025-05-11 21:23:29 [I] nodegroup "my-nodes" has 1 node(s)
2025-05-11 21:23:29 [I] node "ip-192-168-47-54.ec2.internal" is ready
2025-05-11 21:23:29 [I] created 1 managed nodegroup(s) in cluster "my-cluster"
2025-05-11 21:23:31 [I] kubectl command should work with "C:\\Users\\gertr\\.kube\\config", try 'kubectl get nodes'
2025-05-11 21:23:31 [I] EKS cluster "my-cluster" in "us-east-1" region is ready
PS C:\\Users\\gertr> git version
git version 2.49.0.windows.1
PS C:\\Users\\gertr> git clone https://github.com/nspacer/eks-basics
Cloning into 'eks-basics'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 9 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (9/9), 6.09 KiB | 1.01 MiB/s, done.
PS C:\\Users\\gertr>
```

```
Windows PowerShell
PS C:\Users\gertr> cd eks-basics
PS C:\Users\gertr\eks-basics> ls

Directory: C:\Users\gertr\eks-basics

Mode                LastWriteTime         Length Name
----                -
-a----          5/11/2025   9:32 PM           1090 LICENSE
-a----          5/11/2025   9:32 PM           349 nginx-deployment.yaml
-a----          5/11/2025   9:32 PM           166 nginx-svc.yaml
-a----          5/11/2025   9:32 PM            12 README.md

PS C:\Users\gertr\eks-basics> |
```

```
Windows PowerShell
Directory: C:\Users\gertr\eks-basics

Mode                LastWriteTime         Length Name
----                -
-a----          5/11/2025   9:32 PM           1090 LICENSE
-a----          5/11/2025   9:32 PM           349 nginx-deployment.yaml
-a----          5/11/2025   9:32 PM           166 nginx-svc.yaml
-a----          5/11/2025   9:32 PM            12 README.md

PS C:\Users\gertr\eks-basics> cat nginx-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    env: dev
spec:
  replicas: 3
  selector:
    matchLabels:
      env: dev
  template:
    metadata:
      labels:
        env: dev
    spec:
      containers:
        - name: nginx
          image: nginx
          ports:
            - containerPort: 80
PS C:\Users\gertr\eks-basics> |
```

```
Windows PowerShell
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    env: dev
spec:
  replicas: 3
  selector:
    matchLabels:
      env: dev
  template:
    metadata:
      labels:
        env: dev
    spec:
      containers:
        - name: nginx
          image: nginx
          ports:
            - containerPort: 80
PS C:\Users\gertr\eks-basics> cat nginx-svc.yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-svc
  labels:
    env: dev
spec:
  type: LoadBalancer
  ports:
    - port: 80
  selector:
    env: dev
PS C:\Users\gertr\eks-basics> |
```

After cloning the repository containing the Deployment and Service files, both files were applied to the Kubernetes cluster to ensure that the nginx application is deployed and running in the cluster and also ensure that the application is exposed to the public internet or internal network through the Service.

```
Windows PowerShell
PS C:\Users\gertr\eks-basics> kubectl apply -f nginx-deployment.yaml
deployment.apps/nginx-deployment created
PS C:\Users\gertr\eks-basics> ls

Directory: C:\Users\gertr\eks-basics

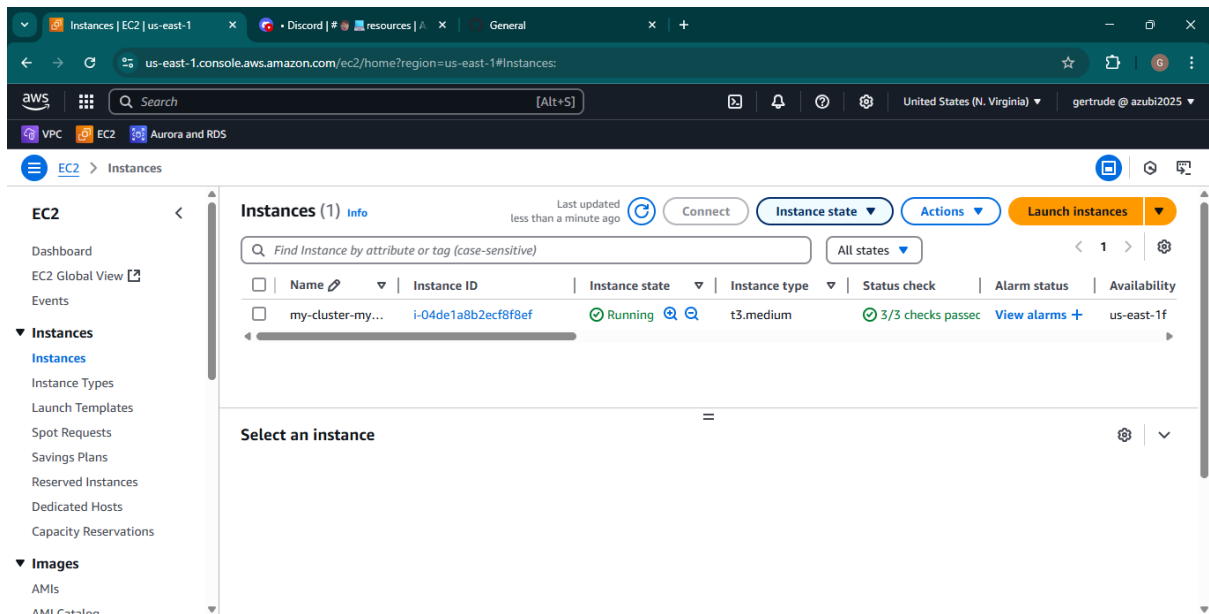
Mode                LastWriteTime         Length Name
----                -
-a----             5/11/2025   9:32 PM           1090 LICENSE
-a----             5/11/2025   9:32 PM           349 nginx-deployment.yaml
-a----             5/11/2025   9:32 PM           166 nginx-svc.yaml
-a----             5/11/2025   9:32 PM            12 README.md

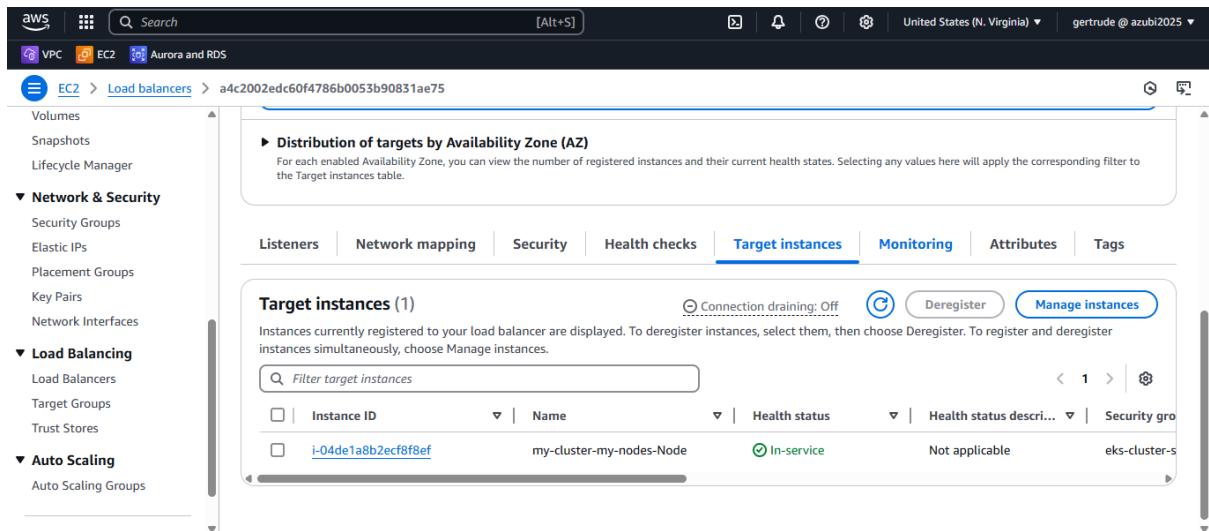
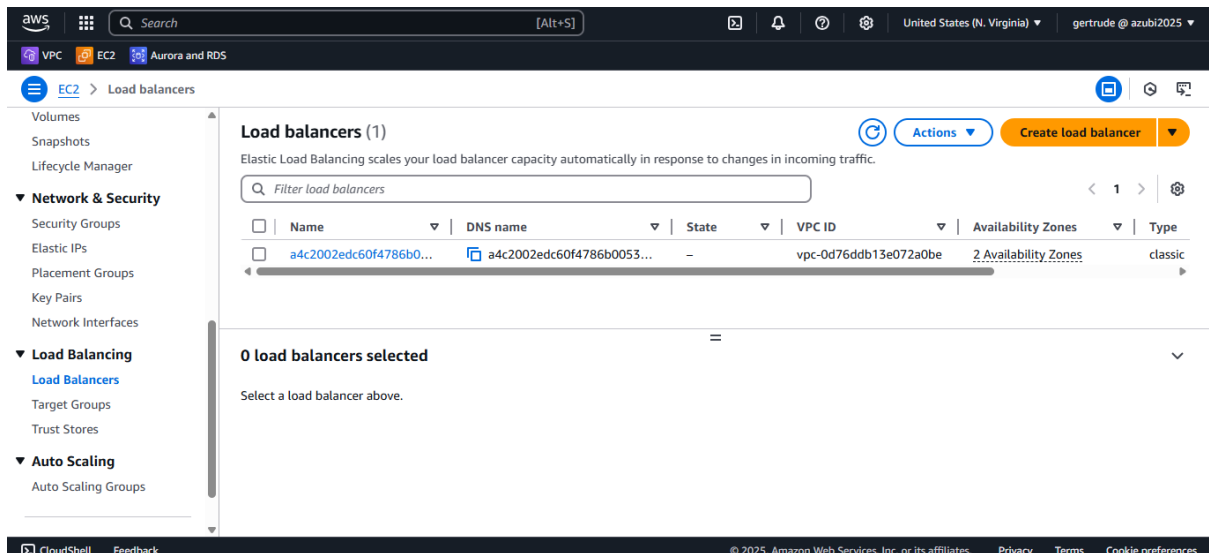
PS C:\Users\gertr\eks-basics> kubectl apply -f nginx-svc.yaml
service/nginx-svc created
PS C:\Users\gertr\eks-basics> |
```

The nodes were ready and the pods were up and running as well. The load balancer was attached to the node as well.

```
Windows PowerShell
PS C:\Users\gertr\eks-basics> kubectl get service
NAME      TYPE        CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
kubernetes ClusterIP   10.100.0.1       <none>           443/TCP      24m
nginx-svc LoadBalancer 10.100.254.155   a4c2002edc60f4786b0053b90831ae75-1719180930.us-east-1.elb.amazonaws.com 80:31810/TCP 94s
PS C:\Users\gertr\eks-basics> kubectl get deployment
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment 3/3     3            3           3m12s
PS C:\Users\gertr\eks-basics> kubectl get nodes
NAME      STATUS   ROLES    AGE   VERSION
ip-192-168-47-54.ec2.internal Ready    <none>  19m   v1.32.3-eks-473151a
PS C:\Users\gertr\eks-basics> kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
nginx-deployment-6b9f544669-45zs8 1/1     Running   0           4m17s
nginx-deployment-6b9f544669-bbzkt 1/1     Running   0           4m17s
nginx-deployment-6b9f544669-wv26b 1/1     Running   0           4m17s
PS C:\Users\gertr\eks-basics>
```

This was verified from the AWS console.





After pasting the DNS of the load balancer on the web browser, this was displayed.

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

I tested cluster availability by stopping nodes and monitoring the status.

The screenshot below shows that the node was shutting down .

The screenshot displays the AWS Management Console interface. At the top, a green banner indicates 'Successfully initiated termination (deletion) of i-04de1a8b2ecf8f8ef'. Below this, the 'Instances' section shows a table with one instance, 'my-cluster-my...', in the 'Shutting-down' state. The instance details for 'i-04de1a8b2ecf8f8ef (my-cluster-my-nodes-Node)' are expanded, showing its Instance ID, Public IPv4 address (3.227.219.95), and Private IPv4 addresses (192.168.47.54 and 192.168.58.217).

Instances (1/1) Info Last updated less than a minute ago Connect Instance state Actions Launch instances

Find Instance by attribute or tag (case-sensitive) All states < 1 > ⚙

<input checked="" type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability
<input checked="" type="checkbox"/>	my-cluster-my...	i-04de1a8b2ecf8f8ef	Shutting-d...	t3.medium	3/3 checks pass	View alarms +	us-east-1f

i-04de1a8b2ecf8f8ef (my-cluster-my-nodes-Node) ⚙ ⌵

Details | Status and alarms | Monitoring | Security | Networking | Storage | Tags

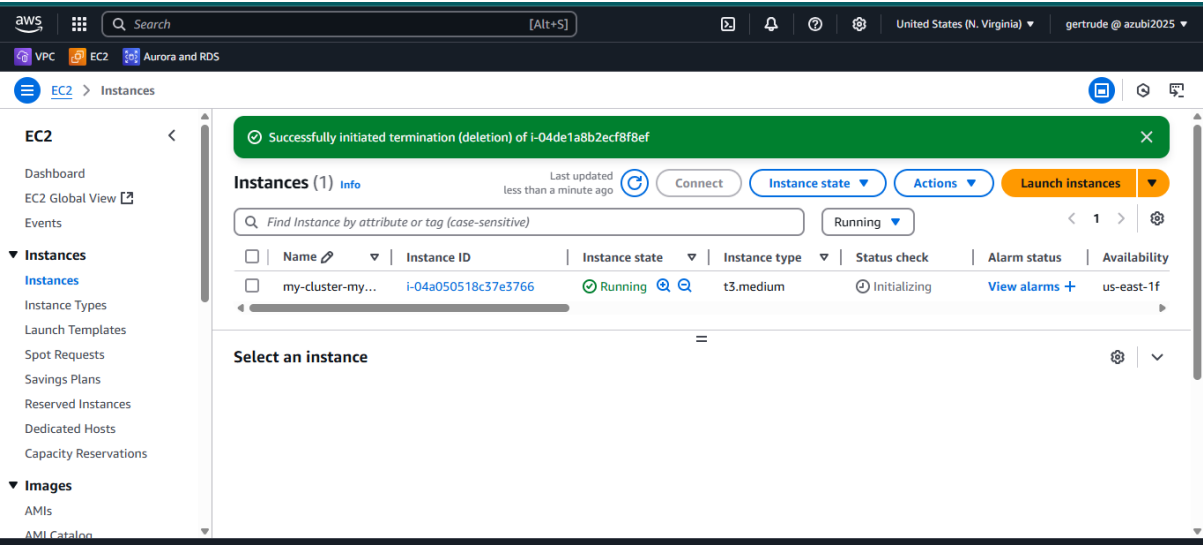
Instance summary Info

Instance ID i-04de1a8b2ecf8f8ef	Public IPv4 address 3.227.219.95 open address	Private IPv4 addresses 192.168.47.54 192.168.58.217
---	---	--

After sometime it was confirmed that there was no resource available meaning the node was successfully terminated.

```
PS C:\Users\gertr\eks-basics> kubectl get service
NAME      TYPE        CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
kubernetes ClusterIP   10.100.0.1       <none>            443/TCP      24m
nginx-svc LoadBalancer 10.100.254.155   a4c2002edc60f4786b0053b90831ae75-1719180930.us-east-1.elb.amazonaws.com 80:31810/TCP 94s
PS C:\Users\gertr\eks-basics> kubectl get deployment
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment 3/3     3            3           3m12s
PS C:\Users\gertr\eks-basics> kubectl get nodes
NAME                                STATUS   ROLES    AGE   VERSION
ip-192-168-47-54.ec2.internal       Ready   <none>    19m   v1.32.3-eks-473151a
PS C:\Users\gertr\eks-basics> kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-6b9f544669-45zs8   1/1     Running   0           4m17s
nginx-deployment-6b9f544669-bbzkt   1/1     Running   0           4m17s
nginx-deployment-6b9f544669-wv26b   1/1     Running   0           4m17s
PS C:\Users\gertr\eks-basics> kubectl get rs
NAME                                DESIRED   CURRENT   READY   AGE
nginx-deployment-6b9f544669         3         3         3       5m9s
PS C:\Users\gertr\eks-basics> kubectl get nodes
No resources found
PS C:\Users\gertr\eks-basics> |
```

After sometime, there was an initiation of the node on the console



And the node was ready after a period as shown below.

```
Windows PowerShell
PS C:\Users\gertr\eks-basics> kubectl get service
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
kubernetes ClusterIP  10.100.0.1       <none>            443/TCP      24m
nginx-svc LoadBalancer 10.100.254.155   a4c2002edc60f4786b0053b90831ae75-1719180930.us-east-1.elb.amazonaws.com 80:31810/TCP 94s
PS C:\Users\gertr\eks-basics> kubectl get deployment
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment 3/3      3            3           3m12s
PS C:\Users\gertr\eks-basics> kubectl get nodes
NAME      STATUS    ROLES    AGE   VERSION
ip-192-168-47-54.ec2.internal Ready     <none>  19m   v1.32.3-eks-473151a
PS C:\Users\gertr\eks-basics> kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
nginx-deployment-6b9f544669-45zs8 1/1     Running   0           4m17s
nginx-deployment-6b9f544669-bbzkt 1/1     Running   0           4m17s
nginx-deployment-6b9f544669-wv26b 1/1     Running   0           4m17s
PS C:\Users\gertr\eks-basics> kubectl get rs
NAME      DESIRED   CURRENT   READY   AGE
nginx-deployment-6b9f544669 3          3         3       5m9s
PS C:\Users\gertr\eks-basics> kubectl get nodes
No resources found
PS C:\Users\gertr\eks-basics> kubectl get nodes
No resources found
PS C:\Users\gertr\eks-basics> kubectl get nodes
NAME      STATUS    ROLES    AGE   VERSION
ip-192-168-40-233.ec2.internal NotReady  <none>  5s    v1.32.3-eks-473151a
PS C:\Users\gertr\eks-basics> kubectl get nodes
NAME      STATUS    ROLES    AGE   VERSION
ip-192-168-40-233.ec2.internal Ready     <none>  68s   v1.32.3-eks-473151a
PS C:\Users\gertr\eks-basics>
```

As expected, when nodes were terminated, Kubernetes detected the loss and automatically provisioned new ones, ensuring high availability and reducing administrative burden.