Deploy NGINX on AWS EKS

STEP 0: Prerequisites

Ensure the following tools are installed:

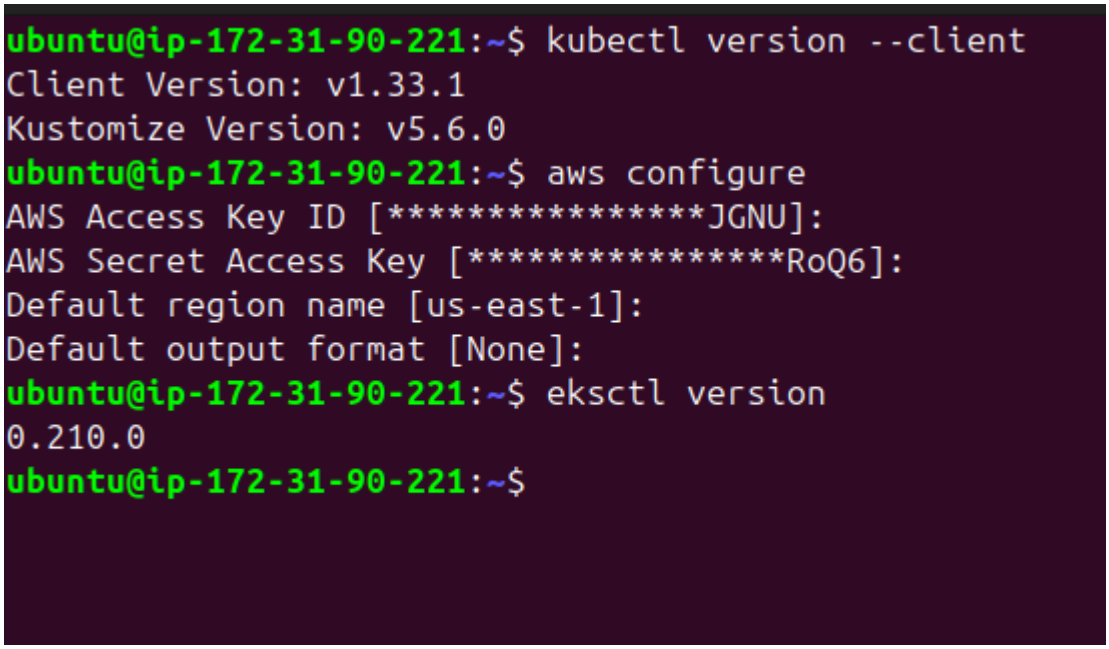
- AWS CLI (https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2.html)

- eksctl (https://eksctl.io/introduction/#installation)

- kubectl (https://kubernetes.io/docs/tasks/tools/)

Configure AWS CLI:

```
aws configure
```

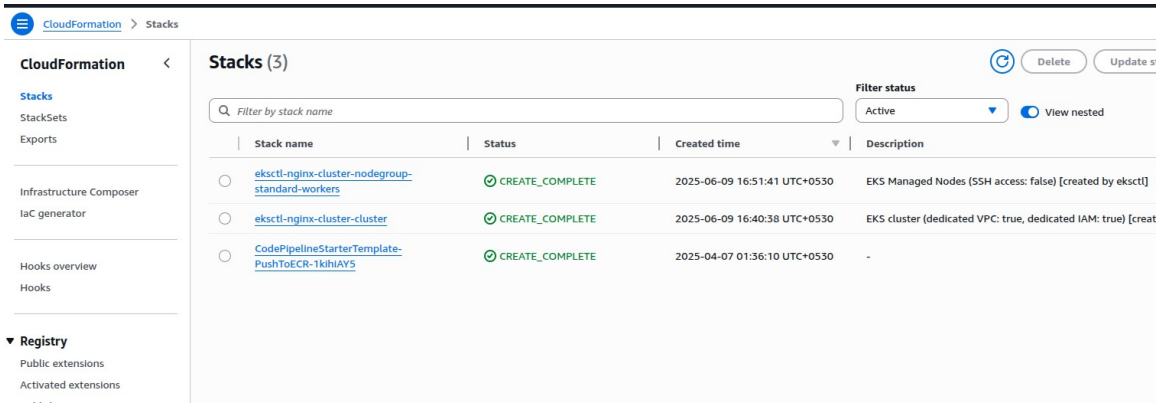


STEP 1: Create EKS Cluster Using eksctl

```
eksctl create cluster \
--name nginx-cluster \
--version 1.29 \
--region us-east-1 \
--nodegroup-name standard-workers \
--node-type t3.medium \
--nodes 2 \
--nodes-min 1 \
--nodes-max 3 \
--managed
```

```
ubuntu@ip-172-31-90-221:~$ eksctl create cluster \
--name nginx-cluster \
--version 1.29 \
--region us-east-1 \
--nodegroup-name standard-workers \
--node-type t3.medium \
--nodes 2 \
--nodes-min 1 \
--nodes-max 3 \
--managed
2025-06-09 11:09:18 [i ]  eksctl version 0.210.0
2025-06-09 11:09:18 [i ]  using region us-east-1
2025-06-09 11:09:18 [!]  Amazon EKS will no longer publ
ich Amazon EKS will release AL2 AMIs. From version 1.33
```

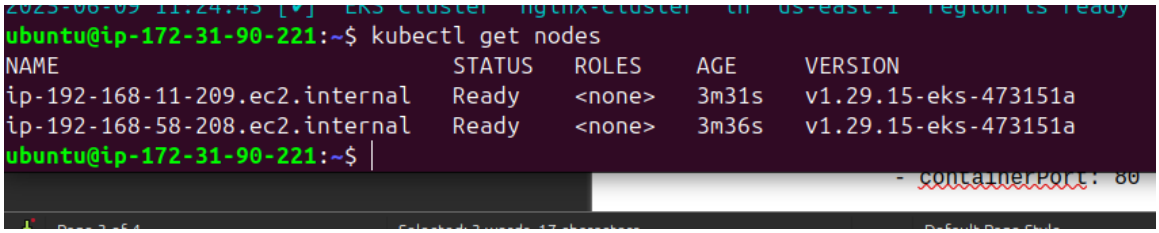
```
2025-06-09 11:24:42 [✔]  saved kubeconfig as "/home/ubuntu/.kube/config"
2025-06-09 11:24:42 [i ]  no tasks
2025-06-09 11:24:42 [✔]  all EKS cluster resources for "nginx-cluster" have been created
2025-06-09 11:24:42 [i ]  nodegroup "standard-workers" has 2 node(s)
2025-06-09 11:24:42 [i ]  node "ip-192-168-11-209.ec2.internal" is ready
2025-06-09 11:24:42 [i ]  node "ip-192-168-58-208.ec2.internal" is ready
2025-06-09 11:24:42 [i ]  waiting for at least 1 node(s) to become ready in "standard-workers"
2025-06-09 11:24:42 [i ]  nodegroup "standard-workers" has 2 node(s)
2025-06-09 11:24:42 [i ]  node "ip-192-168-11-209.ec2.internal" is ready
2025-06-09 11:24:42 [i ]  node "ip-192-168-58-208.ec2.internal" is ready
2025-06-09 11:24:42 [✔]  created 1 managed nodegroup(s) in cluster "nginx-cluster"
2025-06-09 11:24:43 [i ]  kubectl command should work with "/home/ubuntu/.kube/config", try 'kul
2025-06-09 11:24:43 [✔]  EKS cluster "nginx-cluster" in "us-east-1" region is ready
ubuntu@ip-172-31-90-221:~$
```

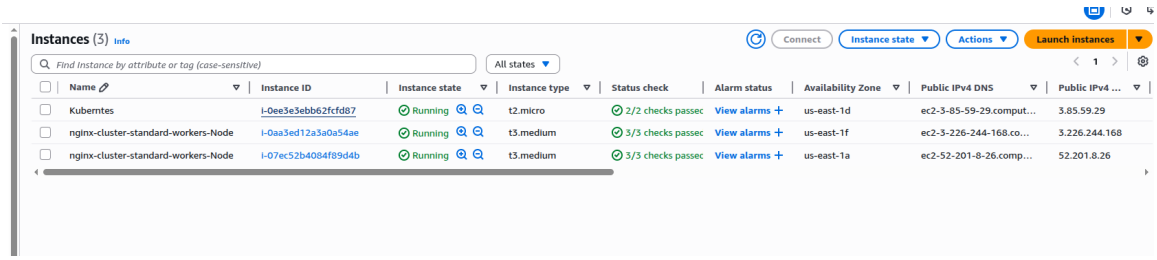


🕒 **Time required:** 10–15 minutes
🎁 **What it creates:** VPC, subnets, EKS control plane, worker nodes

STEP 2: Verify Cluster is Working

```
kubectl get nodes
```





STEP 3: Create NGINX Deployment

Create a file named nginx-deployment.yaml with the following content:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
```

```
kubectl apply -f nginx-deployment.yaml
```

```
kubectl get pods
```

```
ubuntu@ip-172-31-90-221:~$ nano nginx-deployment.yaml
ubuntu@ip-172-31-90-221:~$ kubectl apply -f nginx-deployment.yaml
deployment.apps/nginx-deployment created
ubuntu@ip-172-31-90-221:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-7c79c4bf97-2fsbs   1/1     Running   0          9s
nginx-deployment-7c79c4bf97-tblpw   1/1     Running   0          9s
ubuntu@ip-172-31-90-221:~$
```

STEP 4: Expose NGINX Using LoadBalancer

Create a file named nginx-service.yaml with the following content:

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  type: LoadBalancer
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

```
kubectl apply -f nginx-service.yaml
```

```
kubectl get svc nginx-service
```

```
ubuntu@ip-172-31-90-221:~$ nano nginx-service.yaml
ubuntu@ip-172-31-90-221:~$ kubectl apply -f nginx-service.yaml
service/nginx-service created
ubuntu@ip-172-31-90-221:~$ kubectl get svc nginx-service
NAME            TYPE           CLUSTER-IP       EXTERNAL-IP                                                                PORT(S)        AGE
nginx-service   LoadBalancer   10.100.217.137   a57b9942c204d48328fa9c3ba801e564-1086572734.us-east-1.elb.amazonaws.com   80:32276/TCP   10s
ubuntu@ip-172-31-90-221:~$
```

STEP 5: Access NGINX Web Page

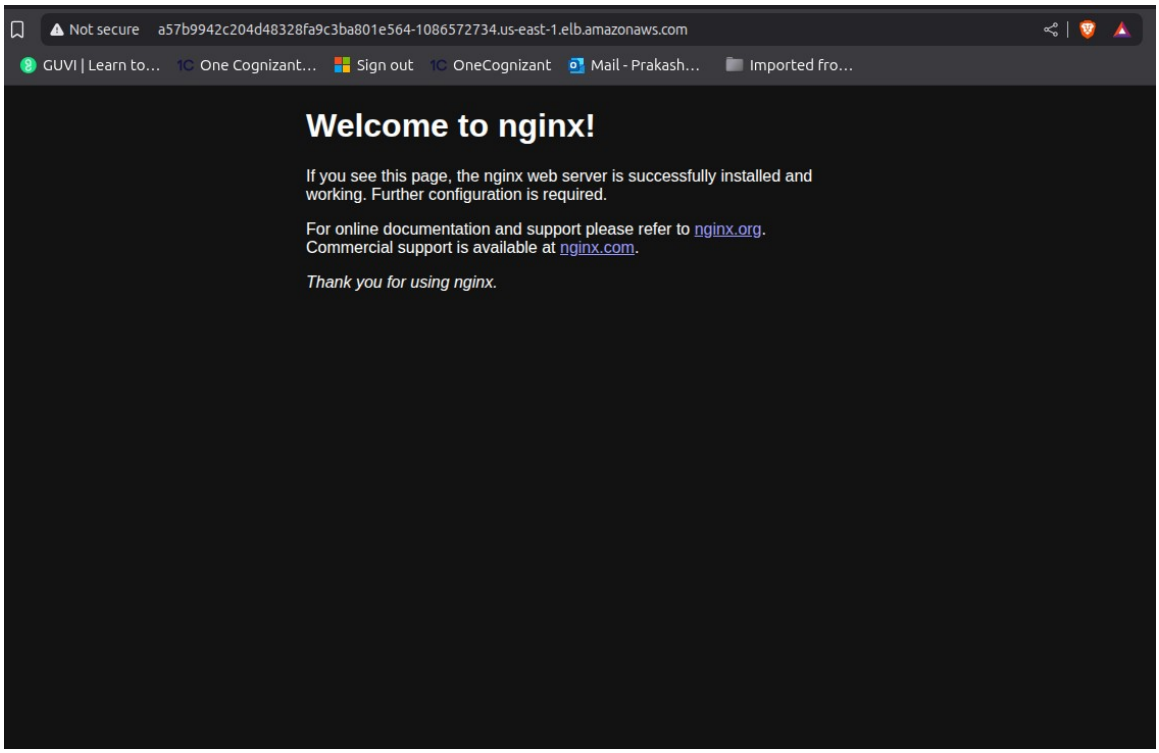
Access the application using the EXTERNAL-IP of the service:

```
curl http://<EXTERNAL-IP>
```

```
ubuntu@ip-172-31-90-221:~$ curl http://a57b9942c204d48328fa9c3ba801e564-1086572734.us-east-1.elb.amazonaws.com
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
ubuntu@ip-172-31-90-221:~$
```



STEP 6: Cleanup (Optional)

```
eksctl delete cluster --name nginx-cluster --region us-east-1
```

```
ubuntu@ip-172-31-90-221:~$ eksctl delete cluster --name nginx-cluster --region us-east-1
2025-06-09 11:37:59 [i]  deleting EKS cluster "nginx-cluster"
2025-06-09 11:37:59 [i]  will drain 0 unmanaged nodegroup(s) in cluster "nginx-cluster"
2025-06-09 11:37:59 [i]  starting parallel draining, max in-flight of 1
2025-06-09 11:37:59 [i]  deleted 0 Fargate profile(s)
2025-06-09 11:37:59 [✔]  kubeconfig has been updated
2025-06-09 11:37:59 [i]  cleaning up AWS load balancers created by Kubernetes objects of Kin
2025-06-09 11:38:25 [i]
2 sequential tasks: { delete nodegroup "standard-workers", delete cluster control plane "ngi
}
2025-06-09 11:38:25 [i]  will delete stack "eksctl-nginx-cluster-nodegroup-standard-workers"
2025-06-09 11:38:25 [i]  waiting for stack "eksctl-nginx-cluster-nodegroup-standard-workers"
2025-06-09 11:38:25 [i]  waiting for CloudFormation stack "eksctl-nginx-cluster-nodegroup-st
2025-06-09 11:38:55 [i]  waiting for CloudFormation stack "eksctl-nginx-cluster-nodegroup-st
```