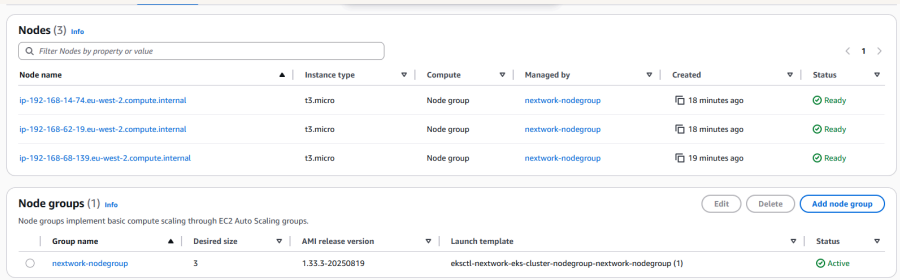
nextwork.org

# Launch a Kubernetes Cluster

Louis Moyo



L
**Louis Moyo**
NextWork Student


# Introducing Today's Project!

In this project, I will create and connect to my first Kubernetes cluster using Amazon EKS, because this is the foundation for deploying containerised applications at scale.

# What is Kubernetes?

Kubernetes is an open-source system for managing containerised applications. Companies and developers use Kubernetes because it automates scaling, load balancing, deployment, and recovery of containers, so apps stay available and resilient without manual effort. It helps save time, improve reliability, and makes it easier to run applications at scale across many servers.

I used eksctl to create a managed Kubernetes cluster on Amazon EKS. The eksctl create cluster command I ran defined the cluster name, AWS region, node type, and number of worker nodes. eksctl handled all of the background work by creating the CloudFormation stacks, VPC networking, IAM roles, and EC2 nodes automatically, so I didn't have to set those up manually.

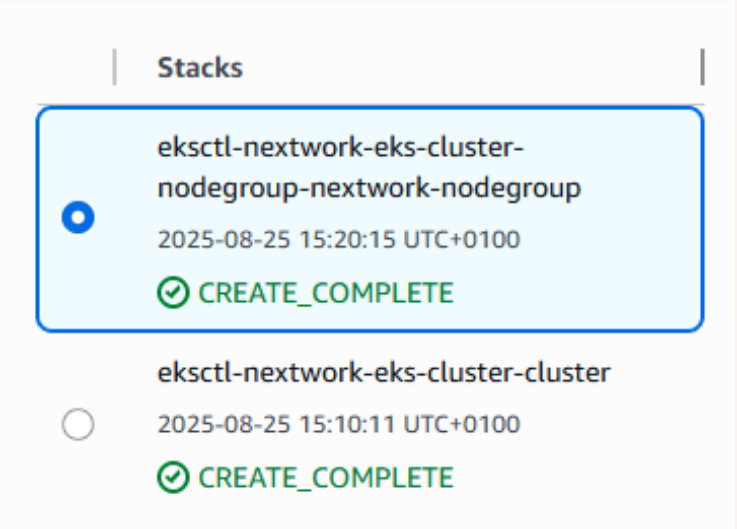
I initially ran into two errors while using eksctl. The first one was because eksctl wasn't installed yet on my EC2 instance, so the command couldn't run. The second one was because my EC2 instance didn't have the right IAM role and policies attached, so it didn't have permission to create the EKS cluster and related resources. Once I fixed both issues, eksctl was able to launch the cluster successfully.


Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023
[ec2-user@ip-172-31-20-113 ~]$ eksctl create cluster \
--name nextwork-eks-cluster \
--nodegroup-name nextwork-nodegroup \
--node-type t3.micro \
--nodes 3 \
--nodes-min 1 \
--nodes-max 3 \
--version 1.33
-bash: eksctl: command not found
[ec2-user@ip-172-31-20-113 ~]$

# eksctl and CloudFormation

CloudFormation helped create my EKS cluster because eksctl uses it in the background to provision all the required resources automatically. It created VPC resources like subnets, route tables, and security groups so that my Kubernetes cluster has a secure network to run in, as well as IAM roles and EC2 worker nodes that the cluster needs to function. Without CloudFormation, I would have had to set up all of these resources manually.

There was also a second CloudFormation stack for the node group that provides the EC2 worker nodes where my Kubernetes pods actually run. The difference between a cluster and a node group is that the cluster contains the control plane (the 'brain' of Kubernetes that manages scheduling and networking), while the node group supplies the worker nodes (the actual servers) that run the containerised applications. Both stacks are needed for the cluster to function.



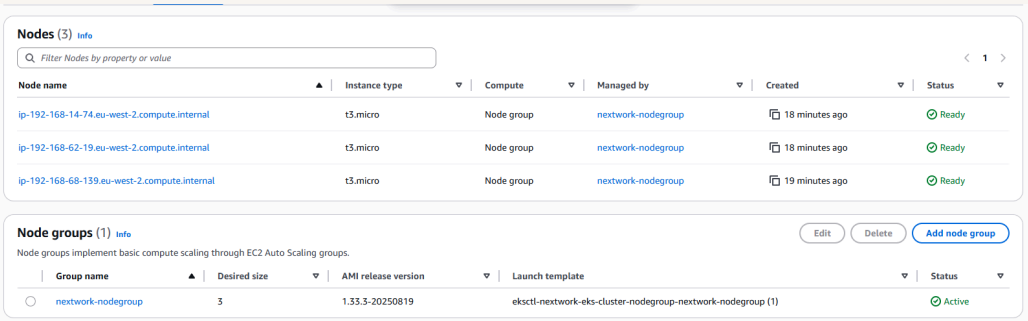
# The EKS console

I had to create an IAM access entry in order to give my IAM user permission to interact with Kubernetes objects in my EKS cluster. An access entry is a way of mapping an IAM principal (like my user account) to Kubernetes RBAC permissions. I set it up by selecting my IAM user's ARN in the EKS console so that my user can run kubectl commands against the cluster.

It took about 15 minutes to create my cluster. Since I'll create this cluster again in the next project of this series, this process could be sped up if I use automation with eksctl or CloudFormation templates, and by reusing VPC and IAM resources instead of creating them from scratch each time.

**Louis Moyo**
NextWork Student





# EXTRA: Deleting nodes

Did you know you can find your EKS cluster's nodes in Amazon EC2? This is because in AWS, every Kubernetes node is actually an EC2 instance that provides the compute power for running pods. Kubernetes calls them 'nodes' in a generic way so it works across different cloud providers, but under the hood in AWS, those nodes are provisioned as EC2 instances.

Desired size means the exact number of nodes I want running in my node group at any given time. Minimum and maximum sizes are helpful for letting Kubernetes and EKS scale the cluster automatically: the minimum size ensures there are always enough nodes to keep the application available, while the maximum size sets an upper limit to control costs and resources. Together, these settings allow Kubernetes to scale up or down based on demand while still maintaining availability and efficiency.

When I deleted my EC2 instances, Kubernetes and EKS automatically launched new nodes to replace them. This is because the node group is managed with a desired state of 3 nodes, so when some nodes were terminated, EKS detected the difference and provisioned new EC2 instances to bring the cluster back to its desired size.

**Louis Moyo**
NextWork Student



# The place to learn & showcase your skills

Check out nextwork.org for more projects