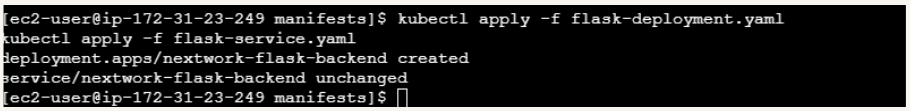
nextwork.org

Deploy Backend with Kubernetes

Louis Moyo



Introducing Today's Project!

In this final project, I'll install kubectl and use it to deploy my backend on Kubernetes, then track the deployment with EKS because this is the final step that takes my container image and manifests from preparation into a running, managed application on the cluster, making my backend live and observable.

Tools and concepts

I used Kubernetes, EKS, ECR, and kubectl to deploy my backend. Key concepts included writing and applying manifest files, understanding Deployments and Services, and managing pods and nodes within an EKS cluster.

Project reflection

This project took me approximately 45 minutes. The most challenging part was fixing YAML validation errors, while my favourite part was seeing the deployment and service successfully created in the cluster.

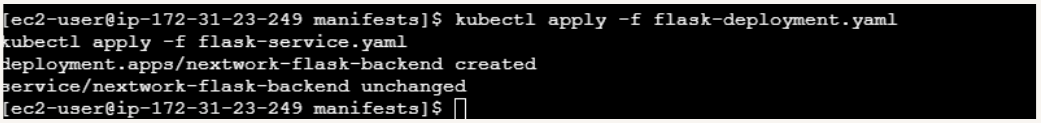
Louis Moyo
NextWork Student

Backend Deployment!

To deploy my backend application, I installed kubectl, then applied my manifest files with kubectl apply -f flask-deployment.yaml and kubectl apply -f flask-service.yaml. This created the Deployment and Service in my EKS cluster, making my backend run and accessible.

kubectl

kubectl is the command-line tool used to interact with Kubernetes resources (like Deployments, Pods, and Services). I need this tool to apply my manifests (e.g., flask-deployment.yaml and flask-service.yaml), which tells Kubernetes to create or update resources inside my cluster. I can't use eksctl for this job because eksctl is mainly for creating, configuring, and deleting EKS clusters. Once the cluster exists, kubectl is the tool that actually manages the workloads (apps, services, scaling, etc.) running inside that cluster.

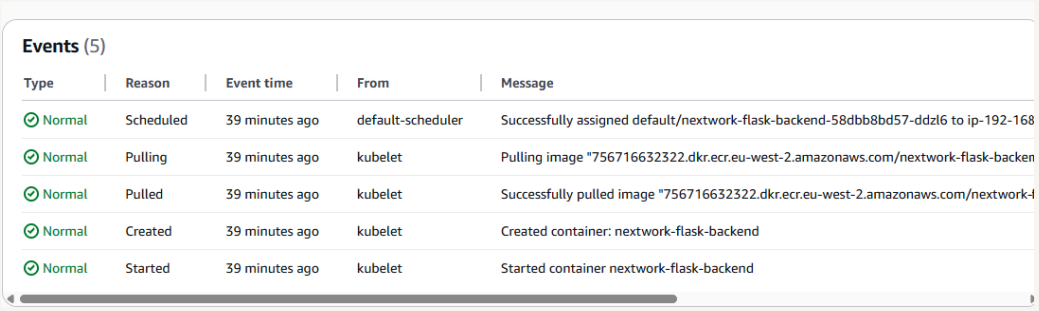

[ec2-user@ip-172-31-23-249 manifests]$ kubectl apply -f flask-deployment.yaml
kubectl apply -f flask-service.yaml
deployment.apps/nextwork-flask-backend created
service/nextwork-flask-backend unchanged
[ec2-user@ip-172-31-23-249 manifests]$

Verifying Deployment

My extension for this project is to use the EKS console to monitor and verify my cluster resources. I had to set up IAM access policies so that kubectl and the console could communicate securely with my cluster. I set up access by attaching the correct IAM role to my EC2 instance and using my IAM user ARN to allow management permissions.

Once I gained access into my cluster's nodes, I discovered pods running inside each node. Pods are the smallest deployable units in Kubernetes, and they bundle one or more containers together so they can work as a unit. Containers in a pod share the same network space and storage, which allows them to communicate and share data more efficiently.

The EKS console shows you the events for each pod, where I could see when the pod was created, scheduled on a node, and when the containers started running. This validated that my backend pods were deployed successfully and that Kubernetes was keeping them healthy by managing restarts if needed.



Events (5)

Type	Reason	Event time	From	Message
Normal	Scheduled	39 minutes ago	default-scheduler	Successfully assigned default/nextwork-flask-backend-58dbb8bd57-ddzl6 to ip-192-168
Normal	Pulling	39 minutes ago	kubelet	Pulling image "756716632322.dkr.ecr.eu-west-2.amazonaws.com/nextwork-flask-backen
Normal	Pulled	39 minutes ago	kubelet	Successfully pulled image "756716632322.dkr.ecr.eu-west-2.amazonaws.com/nextwork-f
Normal	Created	39 minutes ago	kubelet	Created container: nextwork-flask-backend
Normal	Started	39 minutes ago	kubelet	Started container nextwork-flask-backend

The place to learn & showcase your skills

Check out nextwork.org for more projects