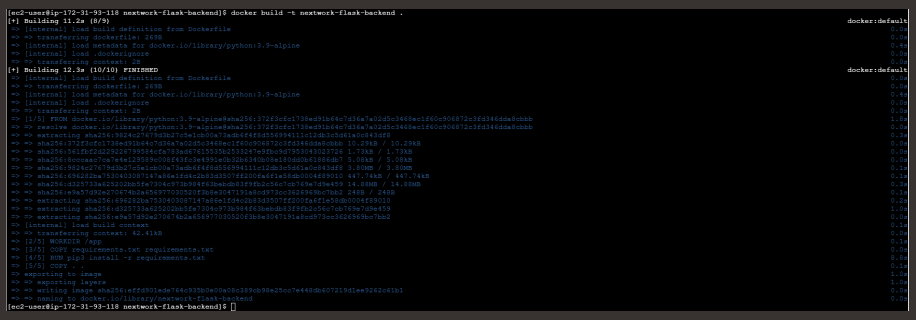
nextwork.org

Set Up Kubernetes Deployment

ME Melvin J Bonner



ME
Melvin J Bonner
NextWork Student

Introducing Today's Project!

In this project, I will pull my app's backend from GitHub and prepare it for deployment with Kubernetes.

Tools and concepts

I used Amazon EKS, EC2, Docker Images, ECR, and Git and Github. Key steps include pulling the code from the backend, building a container image and pushing the image to EKS.

Project reflection

This project took me approximately 90 minutes. The most challenging part was pushing the container image to an ECR repository. My favorite part was using EC2 Instance Connect.

Something new that I learned from this experience was pushing a container image to Amazon ECR.

What I'm deploying

To set up today's project, I launched a Kubernetes cluster. Steps I took to do this included: launching an EC2 instance with the name nextwork-eks-instance, connecting to my instance using EC2 Instance Connect, running the EC2 instance's terminal to download, extract, and install eksctl, attaching an IAM role to the EC2 instance giving it AdministratorAccess, and running the command to create the EKS Cluster.

I'm deploying an app's backend

Next, I retreived the backend that I plan to deploy. An app's backend means is the "brain" of an application. It's where your app processes user requests and stores and retrieves data. Unlike frontend code, which is what users see and interact with, backend code works on the server side to make sure your app behaves as expected when a user does things like clicking on buttons. I retreived backend code by running the command in the terminal to clone my team member's repository.

```
[ec2-user@ip-172-31-93-118 ~]$ git clone https://github.com/NatNextWork1/nextwork-flask-backend.git
Cloning into 'nextwork-flask-backend'...
remote: Enumerating objects: 18, done.
remote: Counting objects: 100% (18/18), done.
remote: Compressing objects: 100% (17/17), done.
remote: Total 18 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (18/18), 6.14 KiB | 2.05 MiB/s, done.
Resolving deltas: 100% (4/4), done.
```

ME
Melvin J Bonner
NextWork Student

Building a container image

Once I cloned the backend code, my next step is to build a container image of the backend. This is because the container image lets Kubernetes set up multiple, identical containers so my application runs consistently across different environments.

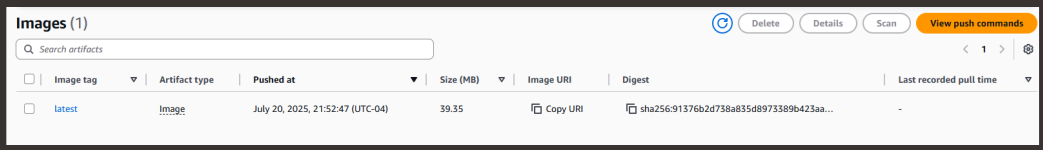
When I tried to build a Docker image of the backend, I ran into a permission error because Docker requires root privileges to function and I'm not currently logged in as the root user.

To solve the permissions error, I added the ec2-user to the docker group. The Docker group is a group in Linux systems that gives users the permission to run Docker commands.

Container Registry

I'm using ECR in this project to store my container image. ECR is a good choice for the job because it's an AWS service, which lets Elastic Kubernetes Service (EKS) deploy my container image with minimal authentication setup.

Container registries like Amazon ECR are great for Kubernetes deployment because my cluster can pull whatever is the latest image in my repository on demand, which makes deployments stay consistent across all nodes automatically.



EXTRA: Backend Explained

After reviewing the app's backend code, I've learned that the app's backend fetches data based on a search topic that's entered, the backend code uses Flask to connect with an external API (Hacker News Search API) and process the data, and the backend then sends the data back as formatted JSON.

Unpacking three key backend files

The requirements.txt file lists flask, flask-restx, requests, and werkzeug. The flask is the web framework used to build the backend code. My app creates an API using an extension Flask-RESTx so that users or other applications can make requests to the backend. The Requests library is used to get data from the Hacker News API. Werkzeug helps Flask handle application-level routing.

The Dockerfile gives Docker instructions on how a Docker image of the backend should be buit. Key commands in this Dockerfile include FROM, LABEL, WORKDIR, COPY, RUN, COPY . ., and CMD.

The app.py file contains the main code for the backend. It has three parts: setting up the app and routing, fetching data, and sending the response.

nextwork.org

The place to learn & showcase your skills

Check out nextwork.org for more projects