

SCD-Assignment-03

Prepared by: Abad Naseer

Submitted to: Laiba Imran

Submission Date

12 December 2024

**Comprehensive Report on Deployment and Management of a Multi-Episode Containerized Application**

**Assignment Details:**

This report documents the deployment and management process for a multi-episode containerized application. The application consists of five services, each with a frontend and backend, developed as separate modules. The deployment was carried out in both Docker and Kubernetes environments as per the assignment's requirements.

**Part 1: Individual Deployment (80 Points)**

**Task Overview**

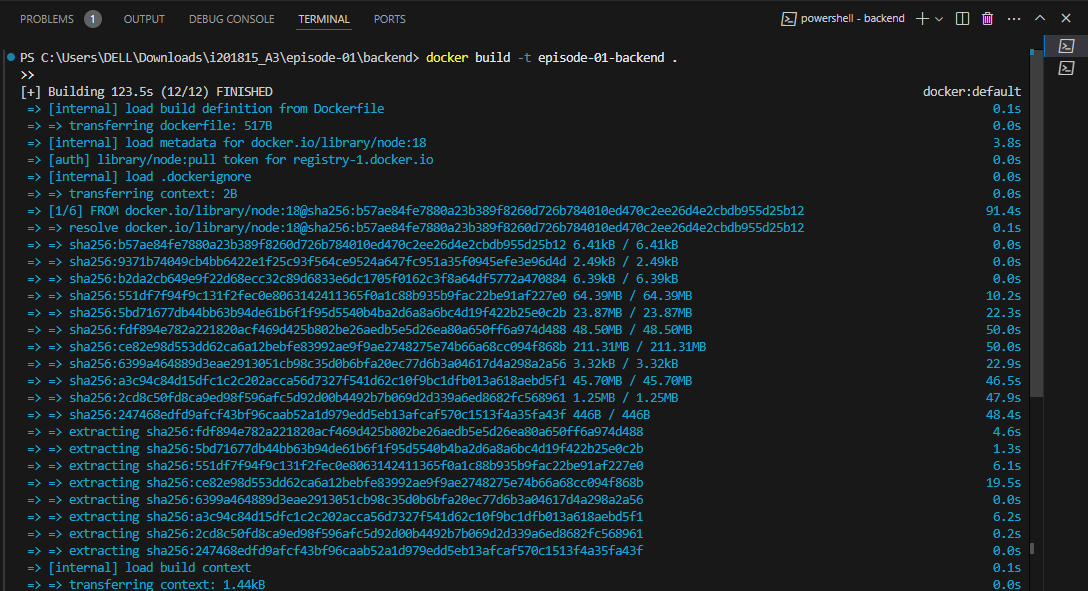
1. **Deploying Individual Modules Locally**
   * Each service's frontend and backend modules were containerized and run locally using Docker.
   * Modules were made accessible and responsive to HTTP requests.

**Dockerfile Details**

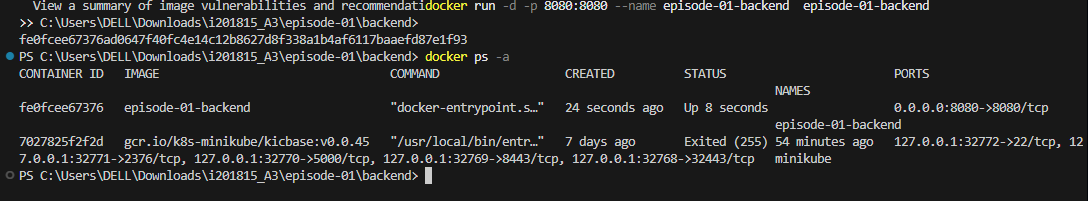
* Each module has its own Dockerfile for frontend and backend.

**Episode-01:**

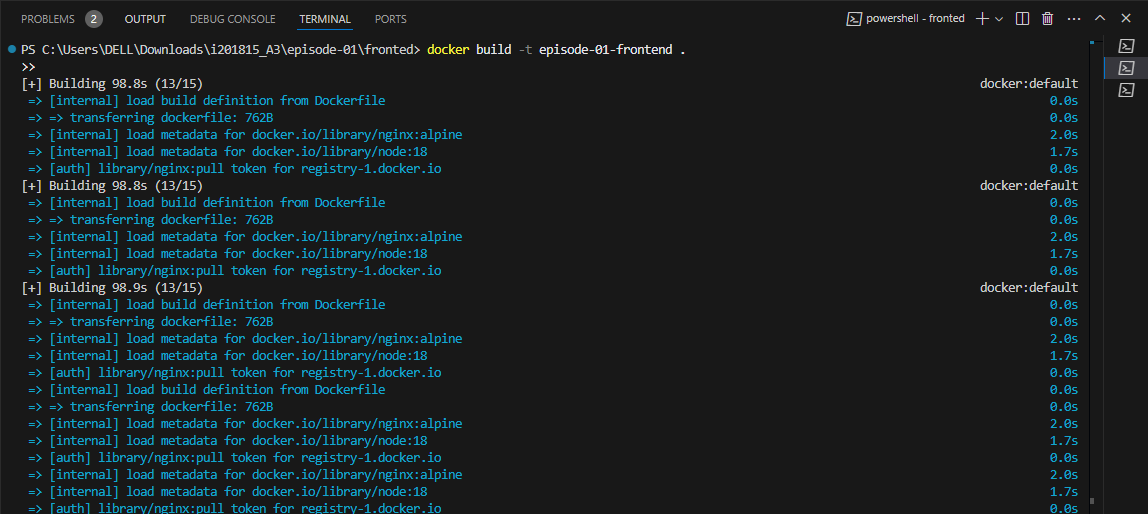
* + **Backend Dockerfile**: Located at C:\Users\DELL\Downloads\i201815\_A3\episode-01\backend\Dockerfile.



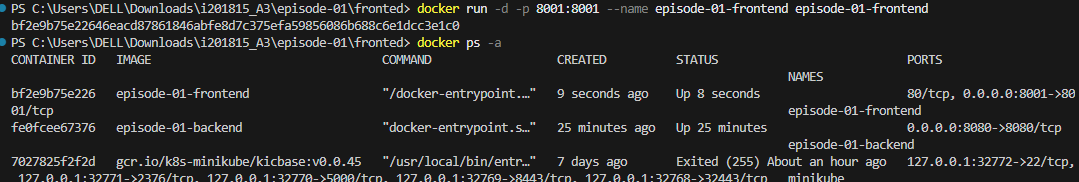
Now below is the command to run docker container for the backend image:



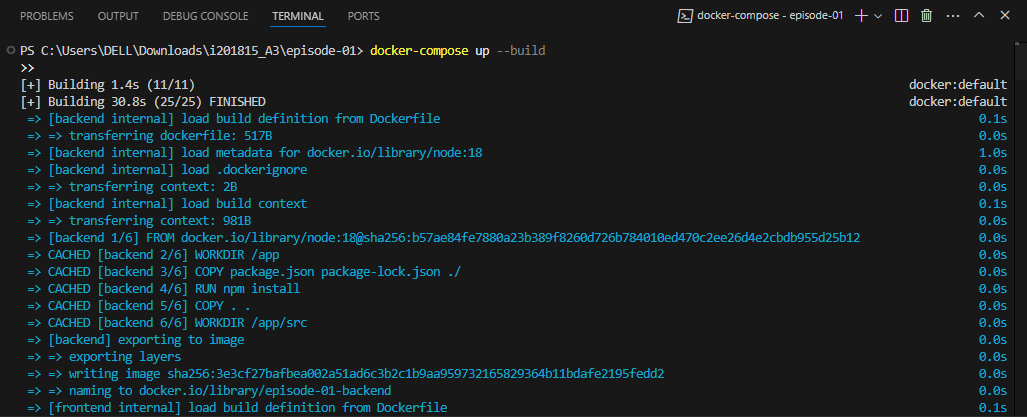
* + **Frontend Dockerfile**: Located at C:\Users\DELL\Downloads\i201815\_A3\episode-01\fronted\Dockerfile.



Here is the screenshot of the frontend service running in a container:



But docker-compose will handle the mongodb part as well by making the communication b/w the frontend, backend and the mongodb for each service. Here is the completed image and running container of the service ( episode-01).



And we can access these routes on localhost:

A screenshot of a computer

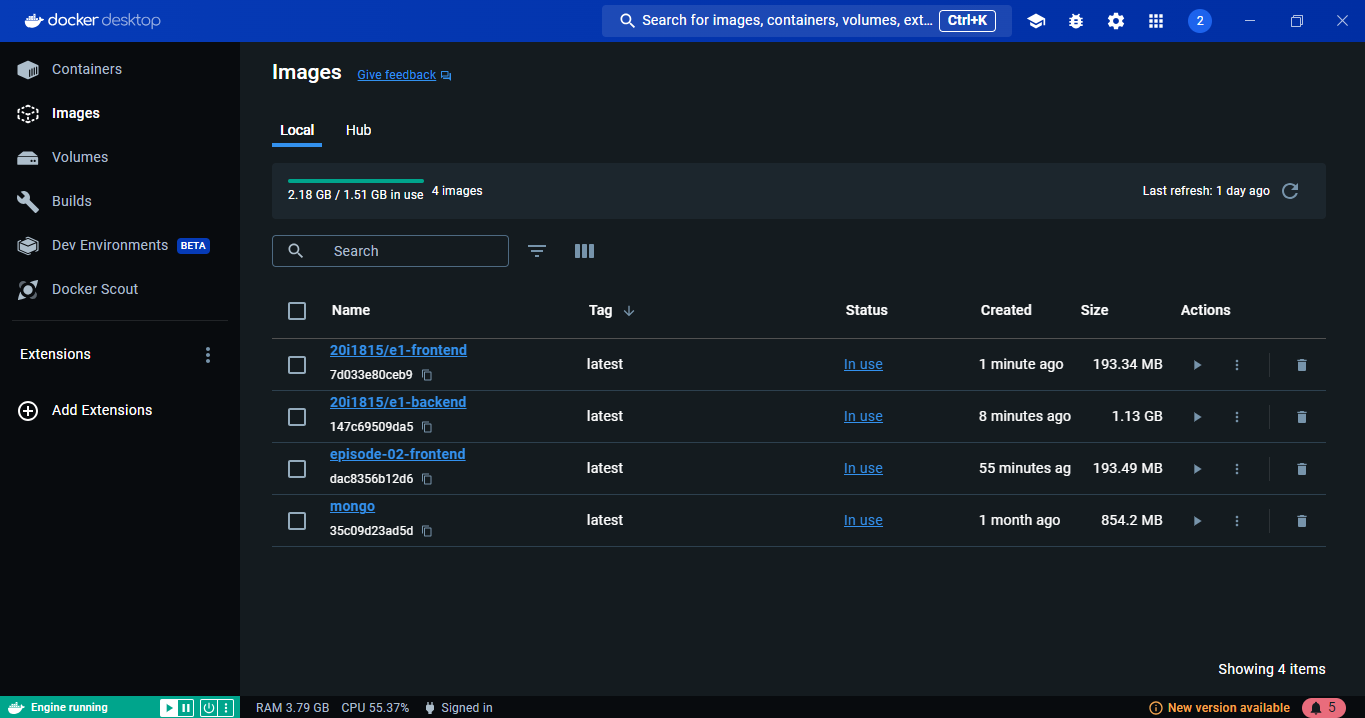
Description automatically generated

And here is the docker desktop containers for episode-01 service running:

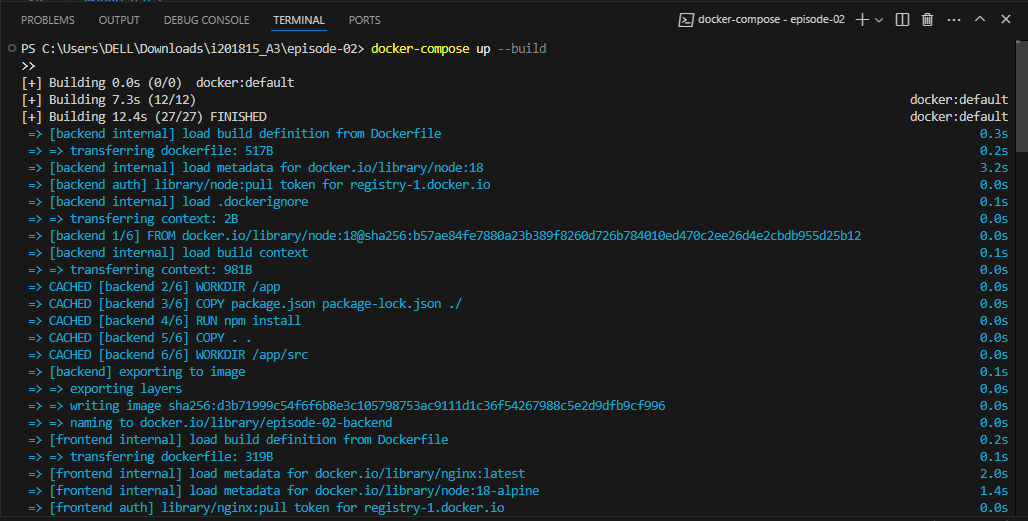
A screenshot of a computer

Description automatically generated

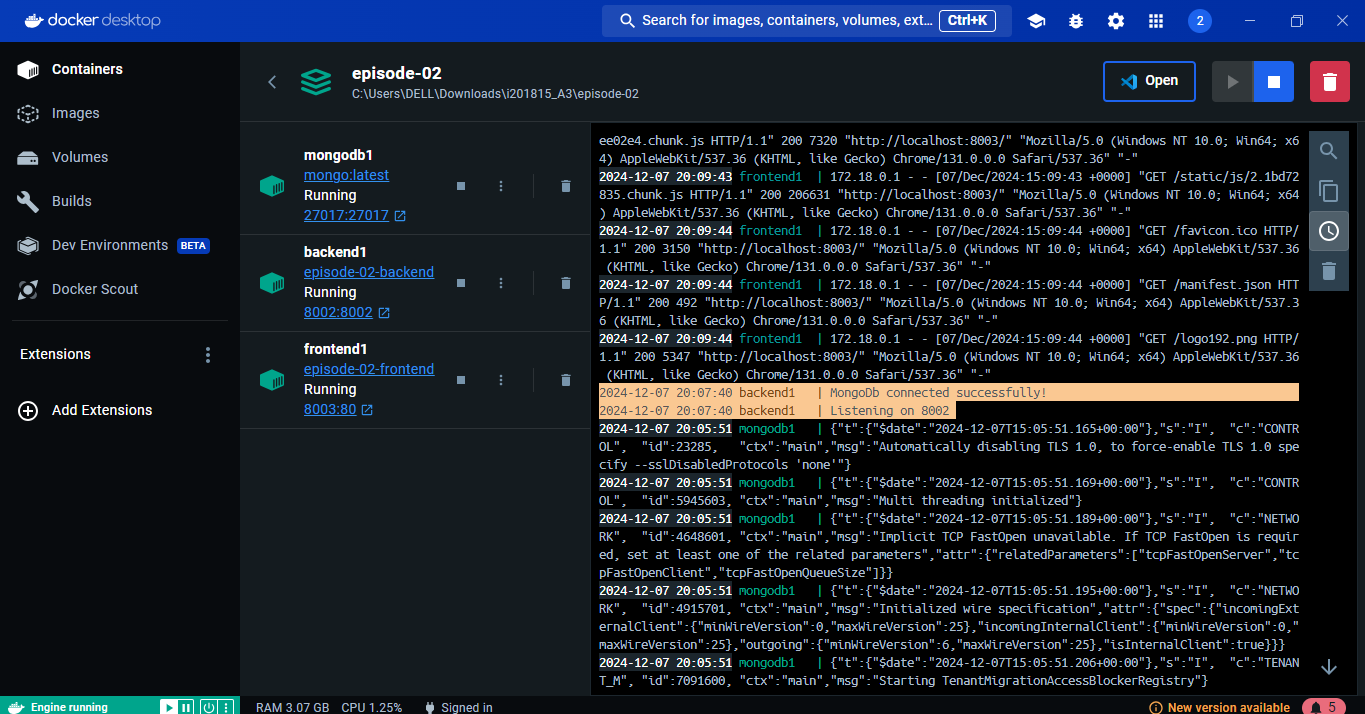
And here are the images:



Episode-02:



And here is the docker desktop log that shows the episode is running perfectly:



And when accessed on browser, it shows the view:

A screenshot of a computer

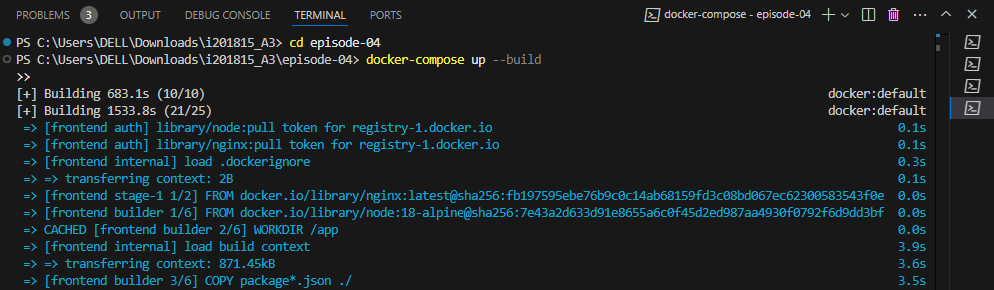
Description automatically generated

And here are the images for episodes-02:

A screenshot of a computer

Description automatically generated

Now here we will find the episode-03:



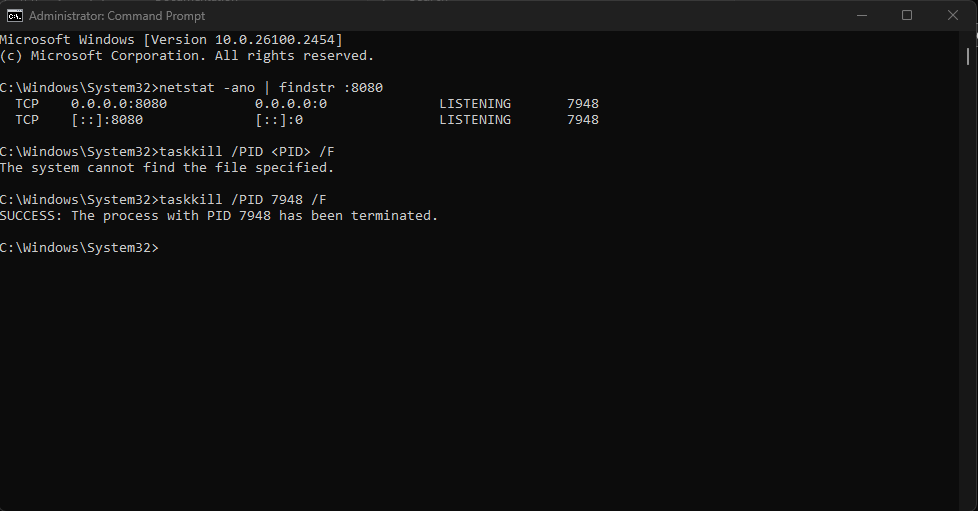
Here you will find image creation and docker container building for episode-05:

A screen shot of a computer

Description automatically generated

**Issues and Resolutions**

* Issue: Port conflicts during initial runs.
  + Resolution: Adjusted the ports in the docker run command.



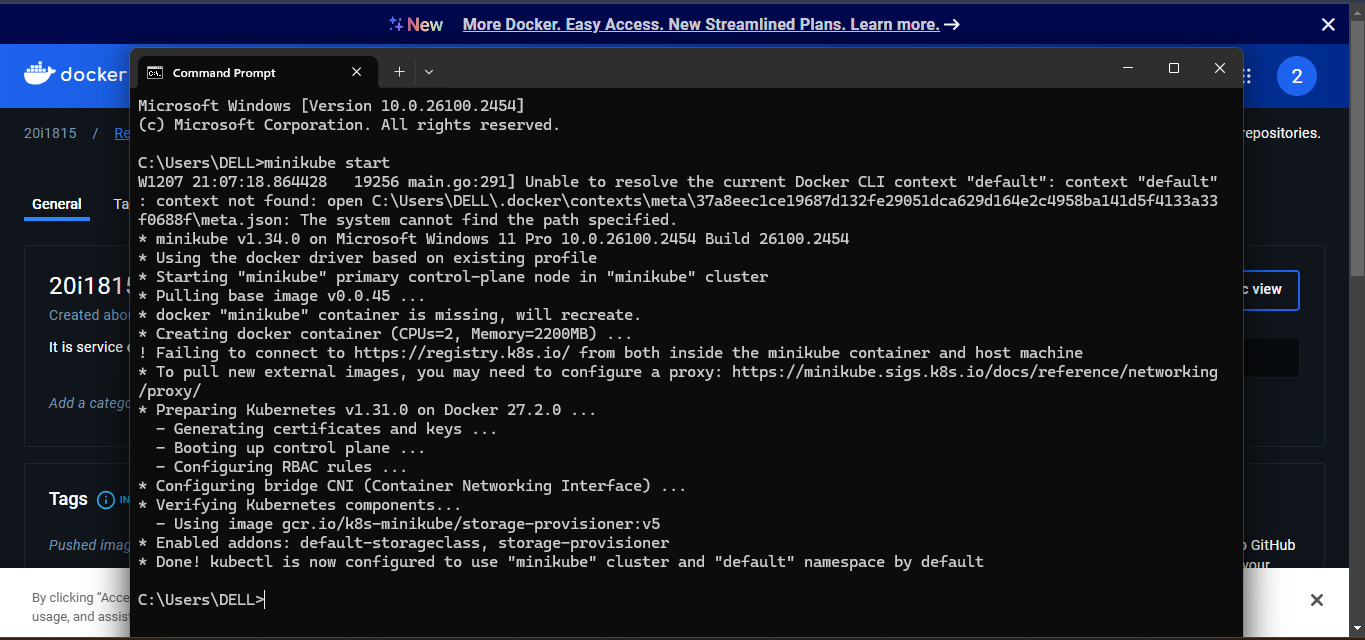
* Issue: Dependency errors during npm install.
  + Resolution: Updated package.json and re-installed dependencies. Error giving me building dependencies so I added this in package.json to resolve ( episode-01-frontend):     "build": "react-scripts --openssl-legacy-provider build",

**Part 2: Kubernetes Deployment (50 Points)**

**Task Overview**

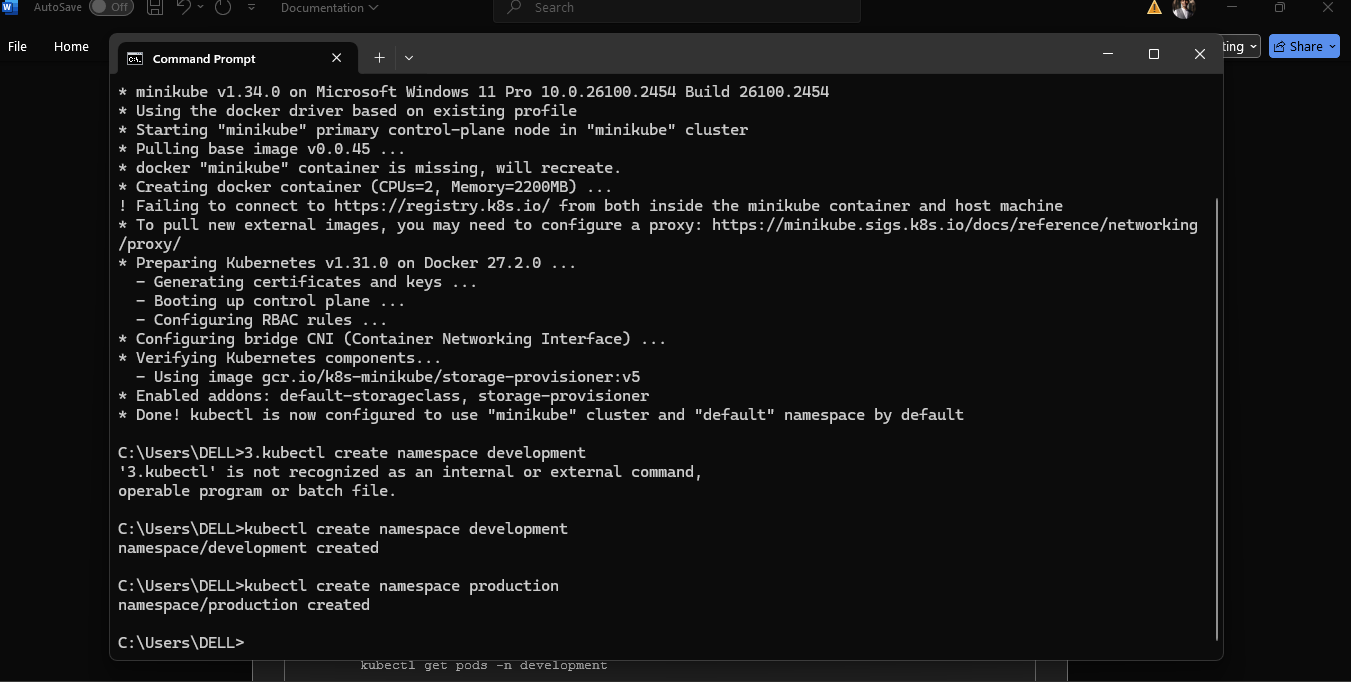
1. **Setting Up a Local Kubernetes Cluster**
   * Minikube was used to set up the cluster.

minikube start



1. **Namespace Configuration**
   * Namespaces were created for development and production environments.
2. kubectl create namespace development

kubectl create namespace production



1. **Module Containerization**
   * Each module's dependencies were encapsulated in their respective containers.

**Part 3: Integration and Final Deployment (20 Points)**

**● Kubectl Deployment:**

**○ Episode 1:**

Kubectl apply -f k8

**○ Episode 2:**

Kubectl apply -f k8

**○ Episode 3:**

Kubectl apply -f k8

**○ Episode 4:**

Kubectl apply -f k8

**○ Episode 5:**

Kubectl apply -f k8

Kubectl apply -f k8

Kubectl apply -f k8

* **Status:**



* **Scaling:**



* **Docker hub Images:**

A screenshot of a computer

Description automatically generated

* **Monitoring:**

minikube -p assignment addons enable metrics-server minikube dashboard –profile=assignment

* **Kubernetes Architecture:**

A diagram of a computer

Description automatically generated

Thank You!