**Week 11 Homework 2: GenAI - Containerized video transcription and chat app**

**Introduction**

In this task, I set up a Streamlit application within a Docker container on Google Cloud Kubernetes. The objective was to configure the necessary environment, including obtaining and using various API keys, building the Docker image, and ensuring the application runs smoothly in the containerized environment. Throughout the process, I focused on overcoming challenges such as validating Docker Compose files, managing environment variables, and accessing container-specific details like IP addresses. By systematically addressing each step, I aimed to achieve a functional and accessible application hosted in the cloud. This project involves creating a chatbot that can answer questions from a video. It showcases a blend of technologies including Docker, OpenAI, Whisper, Embeddings, Chat completions, Pinecone, and Retrieval-Augmented Generation.

**Prerequisites**

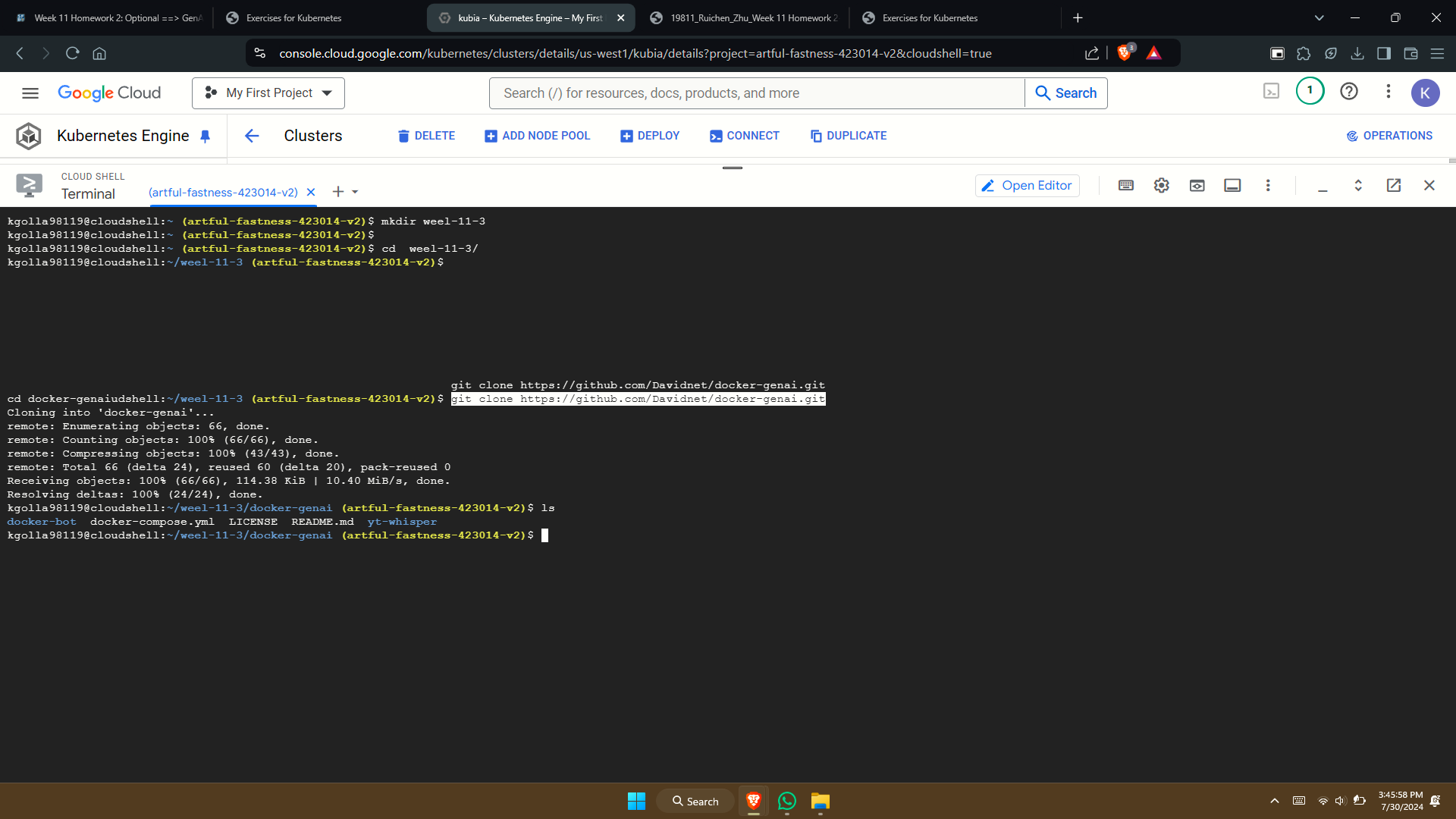
* OpenAI API Key
* Pinecone API Key
* Latest version of Docker Desktop
* GitHub repo

Steps to Build and Run the Application

1. Clone Repository: Clone the project's repository using Git.

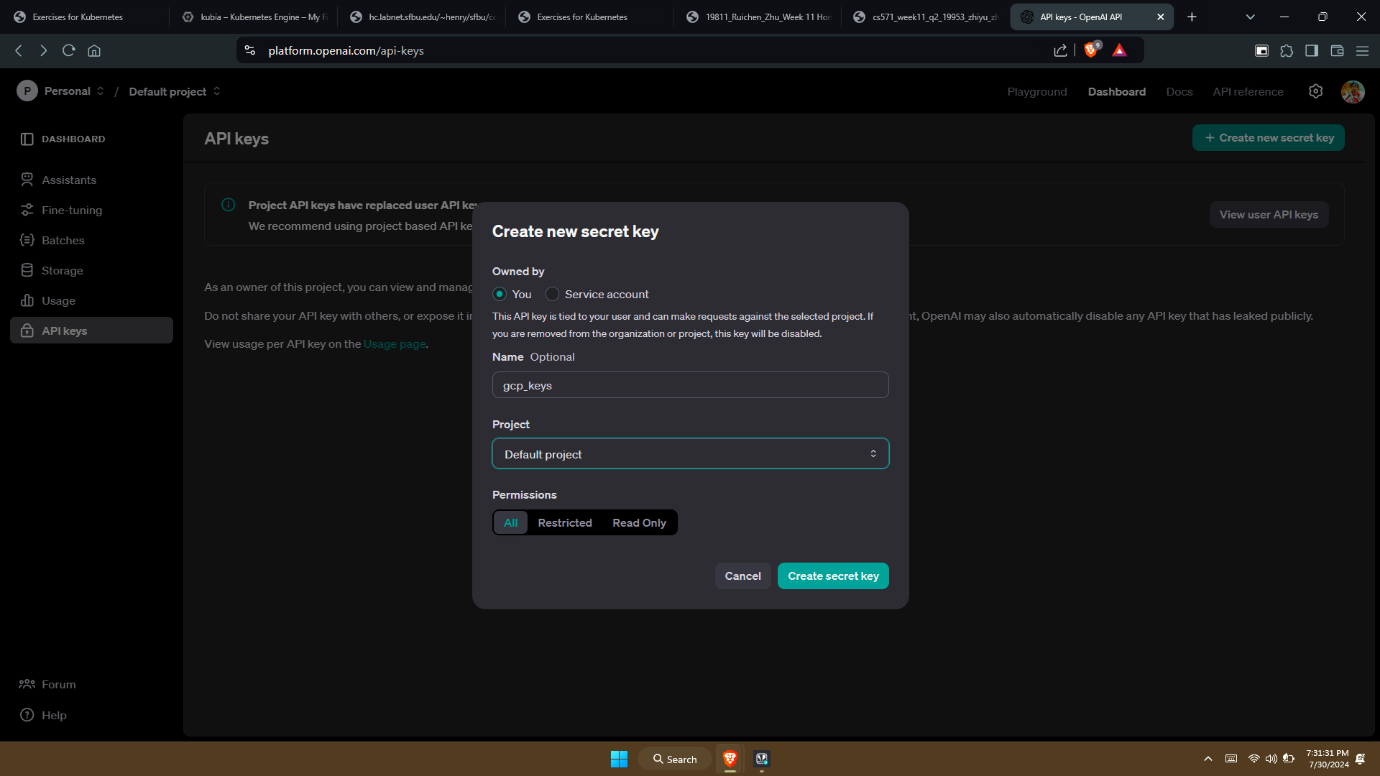
Now we have all the prerequisites we need, we can start with our application. In terminal, go to a desired directory:

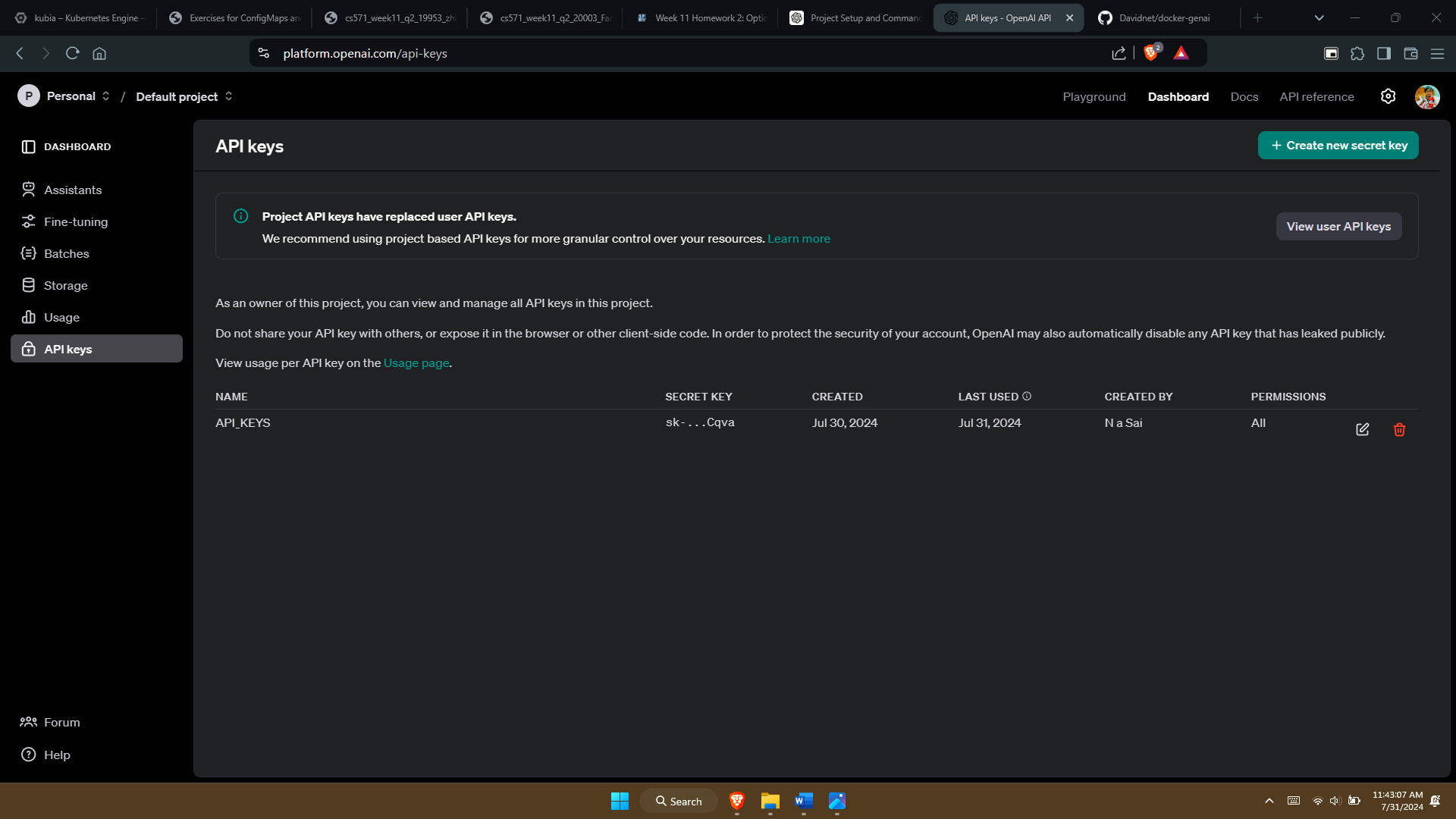
git clone https://github.com/Davidnet/docker-genai.git



Get an OpenAI API Key

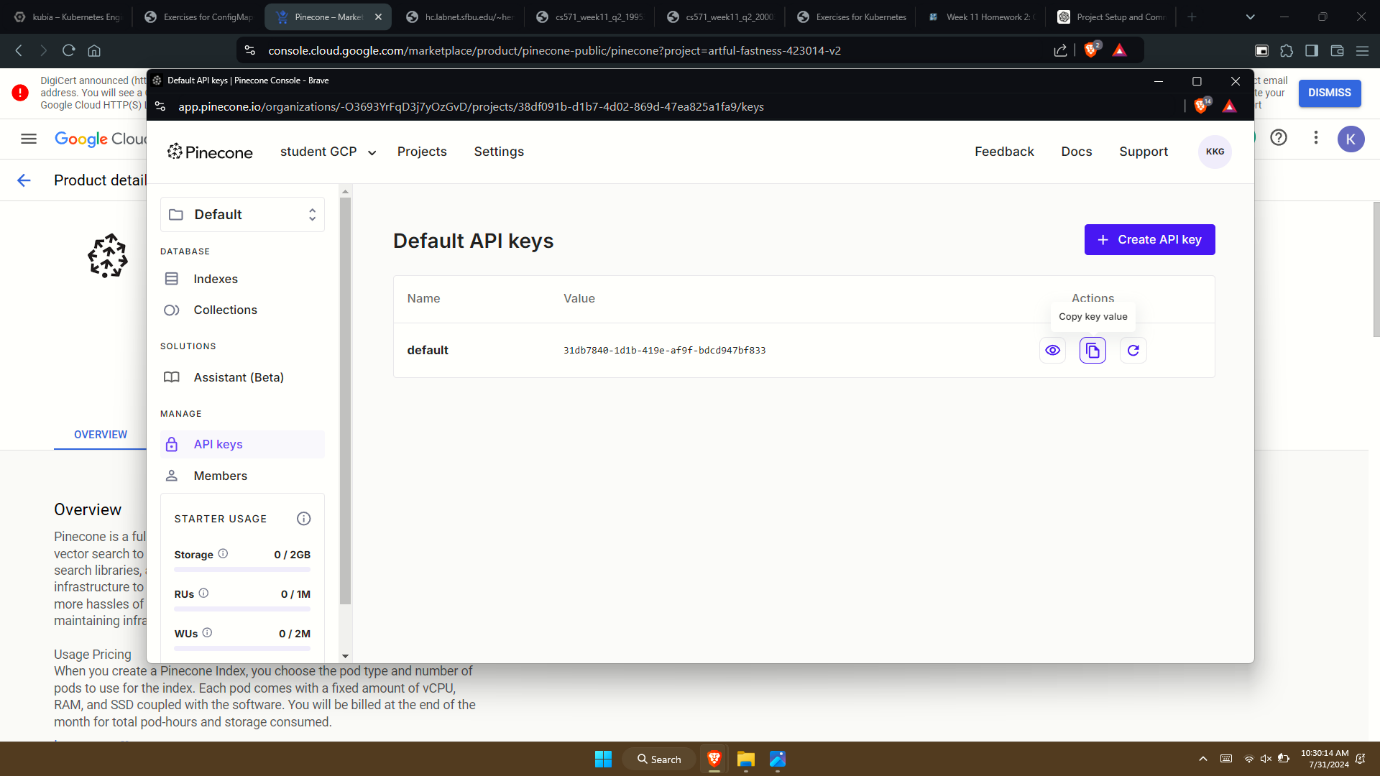
Use this link, log in your OpenAI account, Select “Add payment details” and add some credit. Now that we have enough credit, go to “API keys” from the sidebar menu and select “Create new secret key”. You can choose to give this key a name, then select “Create secret key” to get your API key.





Get a Pinecone API Key

Use this link, log in or create your Pinecone account, then select “Continue with serverless” in the pop-up window, which gives you $100 serverless usage credit. You’ll need Pinecone serverless for this project since our application needs to use Pinecone to create serverless indexes later. Now go to API KEYS from the side bar to get your default API Key.



**Specify your API keys**

Go to docker-genai directory:

cd docker-genai

Create a text file “.env”:

vim .env

Give the below script in the .env file and replace the keys with your Actual keys:

Specify your API keys in “.env” using this template:

#----------------------------------------------------------------------------

# OpenAI

#----------------------------------------------------------------------------

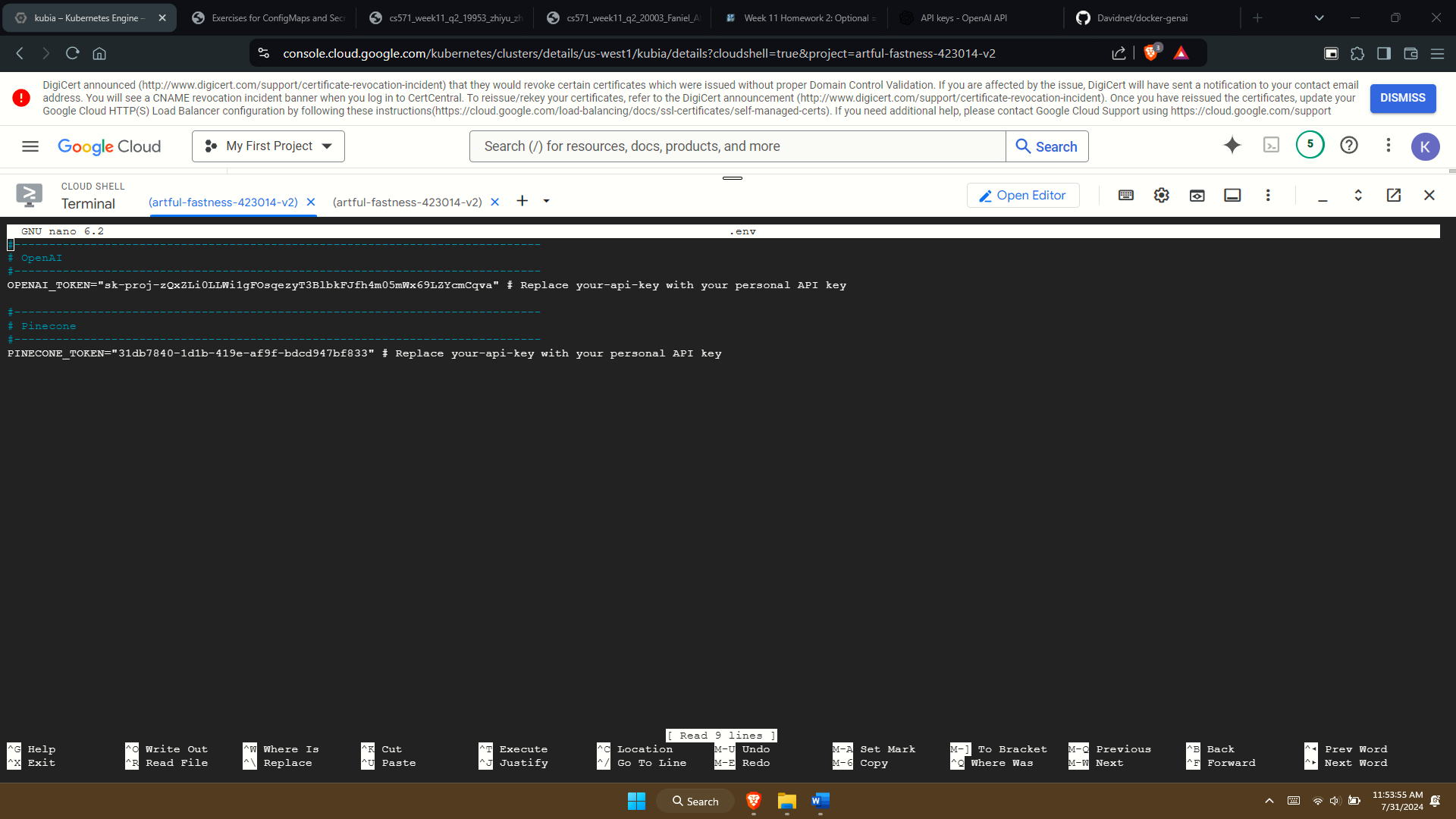
OPENAI\_TOKEN=your-api-key # Replace your-api-key with your personal API key

#----------------------------------------------------------------------------

# Pinecone

#----------------------------------------------------------------------------

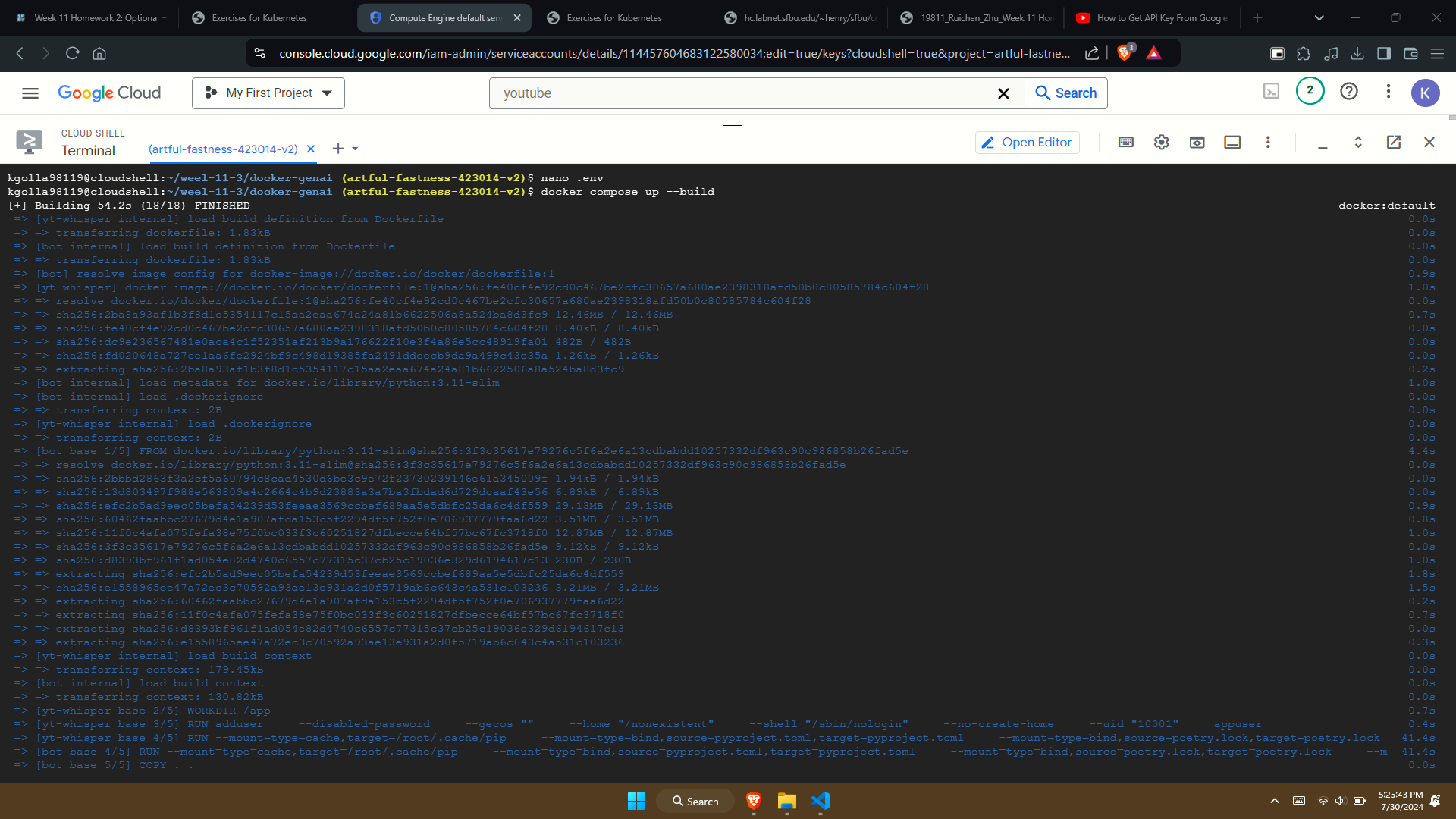
PINECONE\_TOKEN=your-api-key # Replace your-api-key with your personal API key

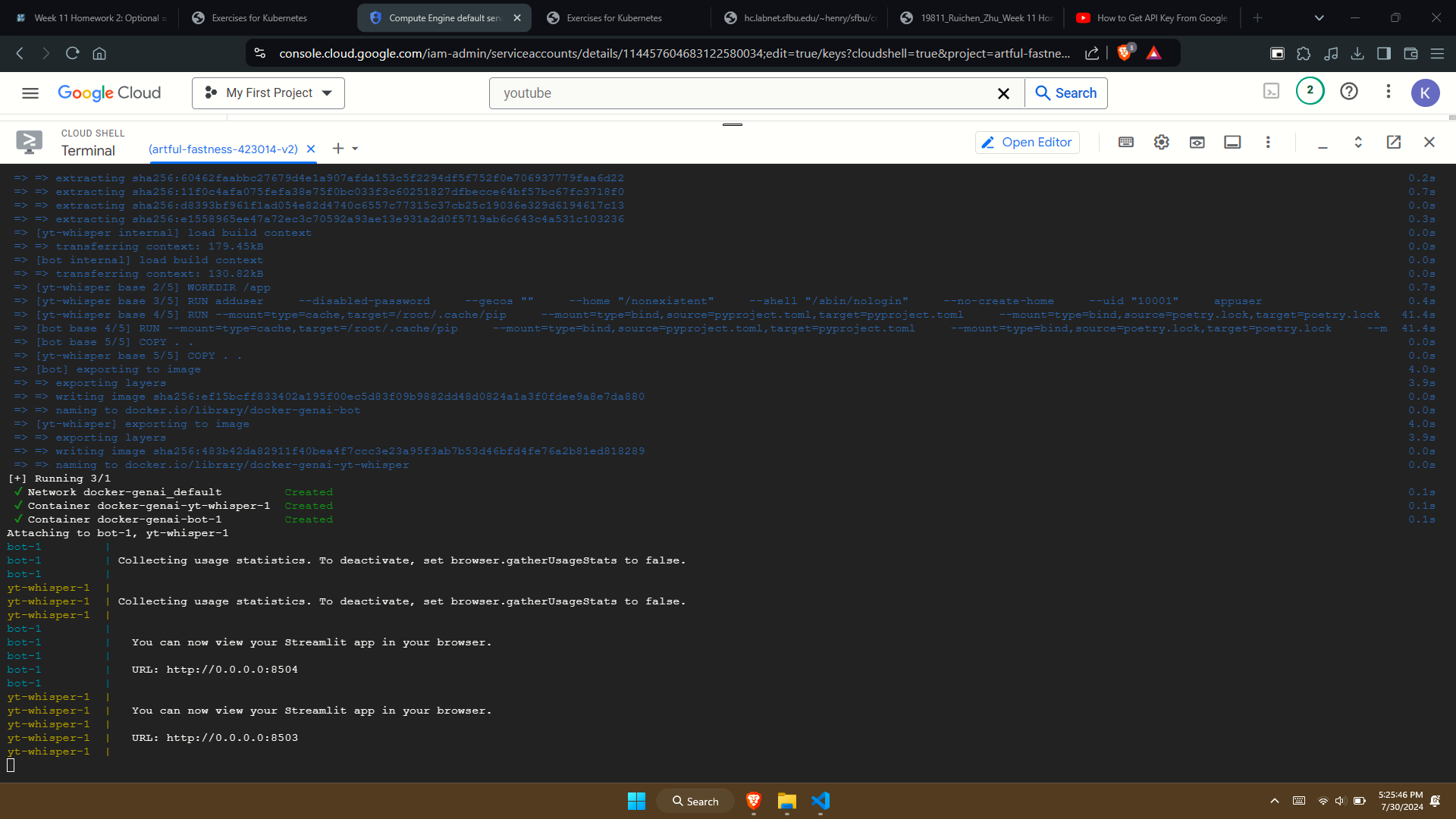


**Build and run the application**

docker compose up –build

You will see something like this while Docker is building the application, Once the application is running, you'll see something like this

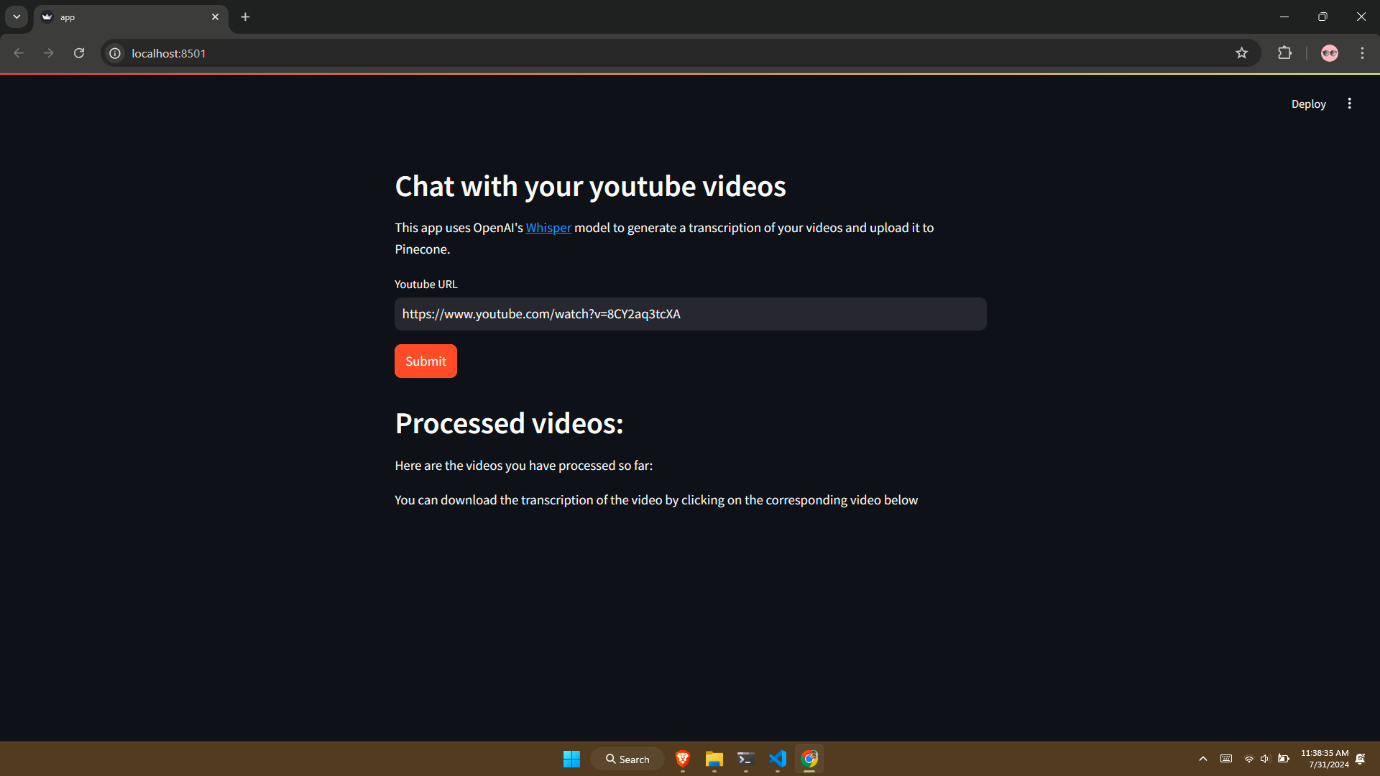


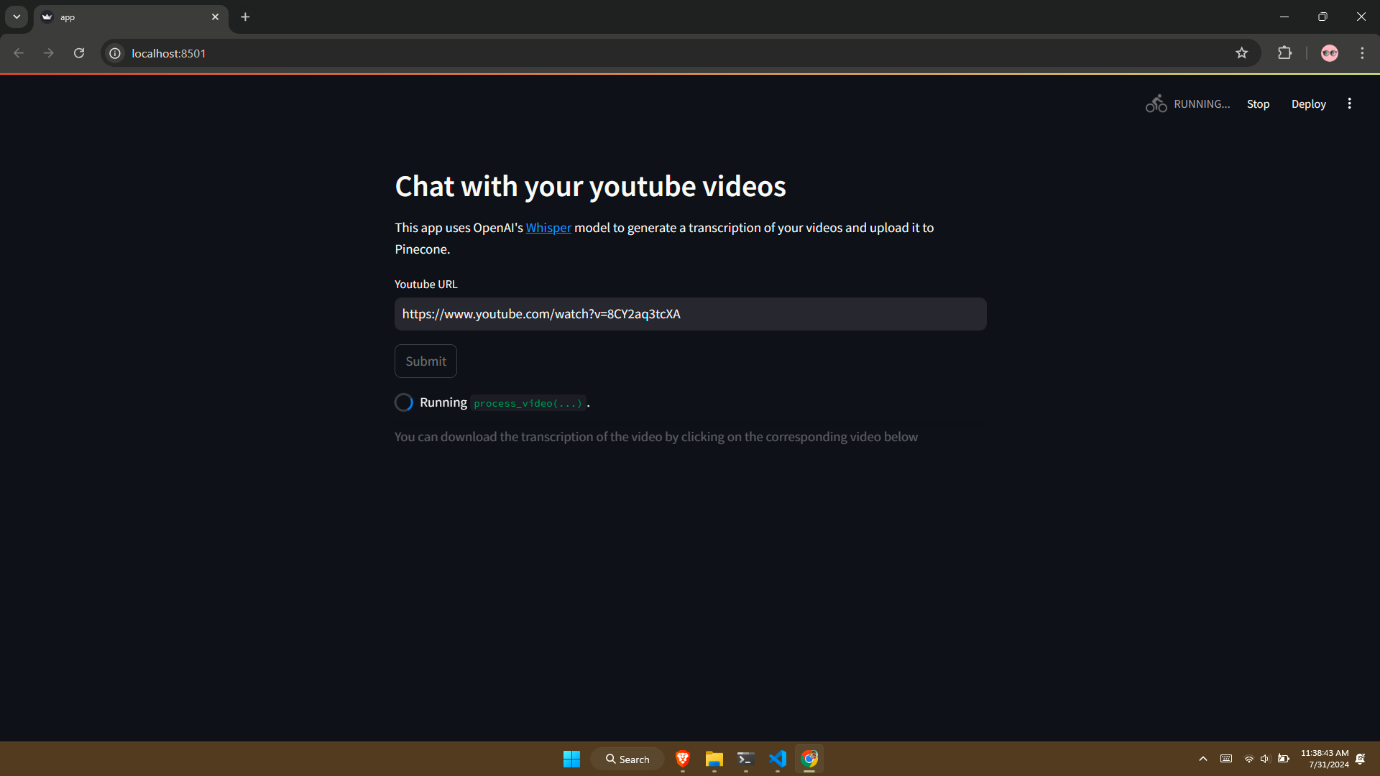


**Use the yt-whisper service**

Open a browser and access the yt-whisper service at http://localhost:8503.

Enter the Youtube video URL you want to use and select “Submit”:





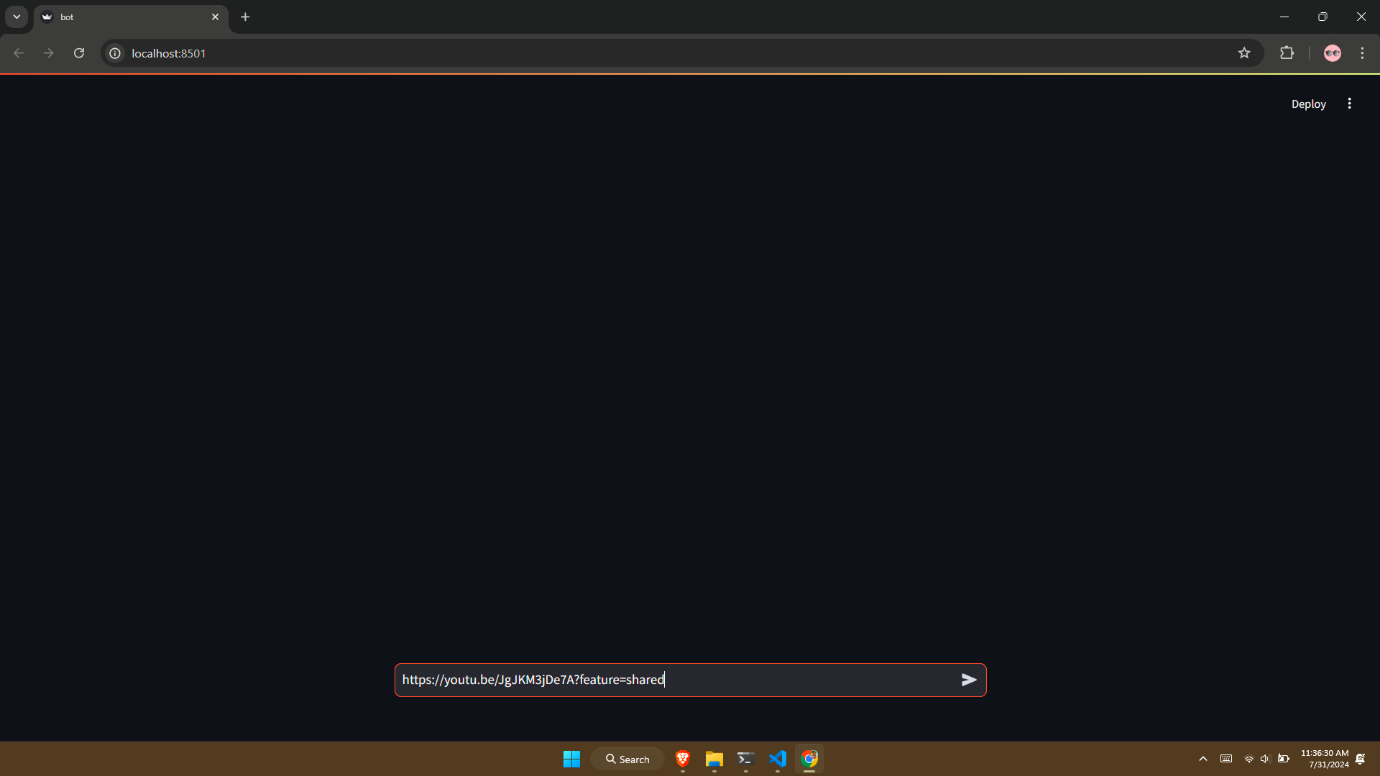


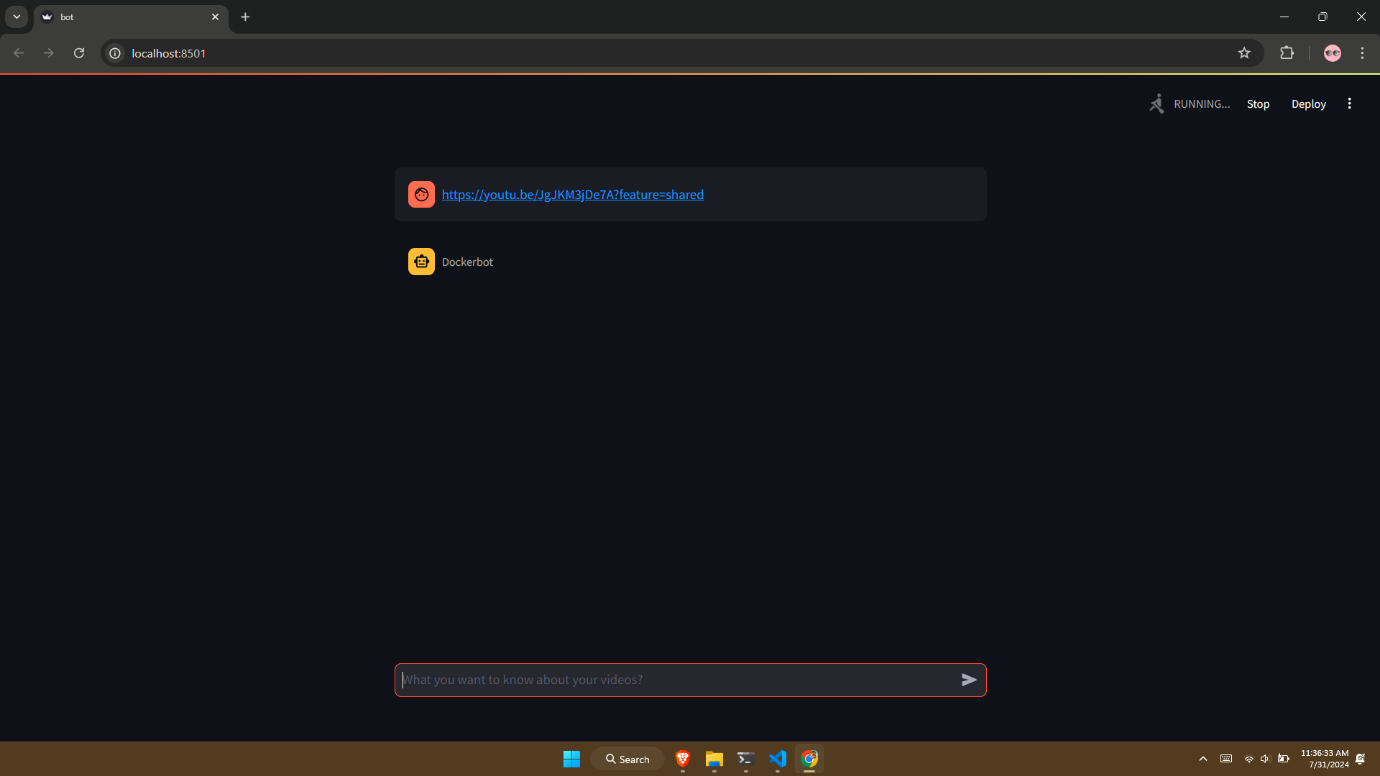
**Use the dockerbot service**

Open a browser and access the dockerbot service at http://localhost:8504.

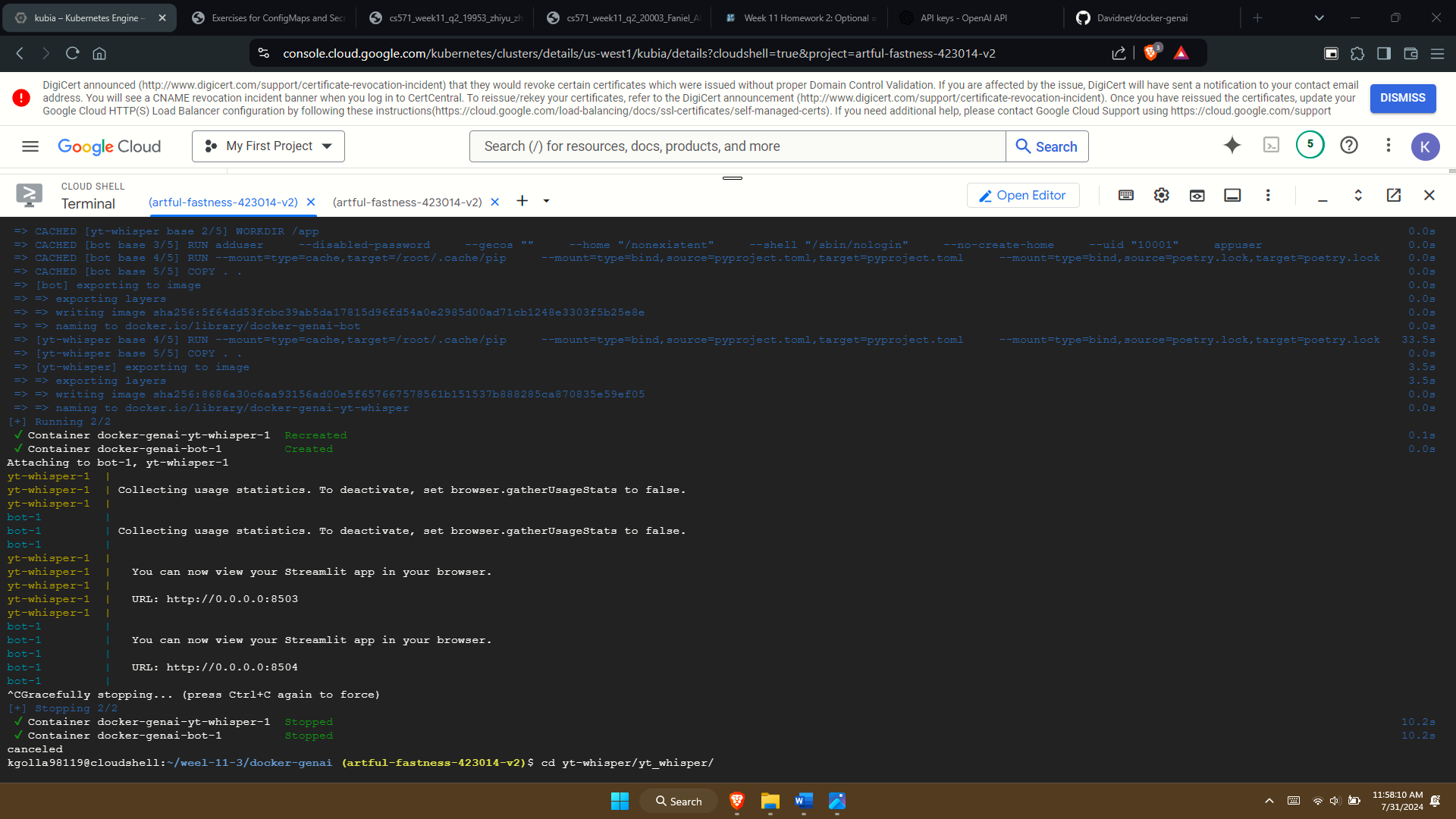
Enter the question you want to ask about the video that was processed.

Dockerbot answers the question and provides links to the video with timestamps:





**Stop the application**

In terminal, press ctrl + C to stop the application: 

**Conclusion**

By the end of this task, I successfully configured and deployed my Streamlit application in a Docker container on Google Cloud Kubernetes. I learned how to handle environment variables securely, use Docker commands to manage and troubleshoot containers, and integrate external APIs to enhance the application's functionality. This experience not only deepened my understanding of containerization and cloud deployment but also equipped me with practical skills for managing complex cloud-based applications. I am now more confident in my ability to deploy and maintain robust applications in a cloud environment.