A picture containing design

Description automatically generated with low confidence

Professor : Maria Fazio

Arya Khosravirad

Matricola: 534 170

A grey logo on a black background

Description automatically generated with low confidence

A picture containing text, font, logo, symbol

Description automatically generated

*Deploying Function with Knative: Automated Video to Image Converter*

*CONTENTS*

1. Introduction
2. Commands
3. Errors
4. Refferences

INTRODUCTION

This project is about creating a tool that changes videos into pictures. The tool lets users upload a video and then splits it into individual frames. We use Docker to package the tool. The tool uses Flask to handle uploads and ffmpeg to process the video. We set up a local Kubernetes cluster with Minikube to test everything. The tool is deployed using Kubernetes and Knative, which helps manage and scale the tool. We tested the tool by uploading videos through a web page and checking that the pictures are created correctly. This project shows how to build, package, deploy, and manage a tool to make video to picture conversion easy and efficient.

1. This script sets up an Ubuntu system on WSL2. It creates a new user account, updates the system packages, and installs the tools that we need for the project. We esure the system is ready for development tasks.

 Update the package lists from repositories:

sudo apt update

 Upgrade installed packages to the latest versions:

sudo apt upgrade

 Install essential development tools:

sudo apt install curl wget git build-essential

 Verify package lists are up to date(final update command).

sudo apt update

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A computer screen with white text

Description automatically generated

A screenshot of a computer

Description automatically generated

1. In this step, we used Windows PowerShell to manage WSL2. We listed the available WSL distributions, updated WSL, set the default WSL version, and ran updates and upgrades inside the WSL environment. This makes sure that both WSL and the Ubuntu system have the latest software and tools.

 List all installed WSL distributions with detailed information:

wsl --list --verbose

 Check for and install updates for WSL.

wsl --update

 Set WSL2 as the default version:

wsl --set-default-version 2

 Update the packages within wsl:

wsl sudo apt update

 Upgrade packages within WSL:

wsl sudo apt upgrade -y

 List WSL distributions again:

wsl --list --verbose

A screenshot of a computer

Description automatically generated

1. In this step, we installed Docker on Ubuntu in WSL2. We updated package lists, installed necessary packages, added Docker's GPG key and repository, updated package lists again, checked if Docker was running, added the user to the Docker group, and ran a test Docker container to make sure it works.

 Update package lists:

sudo apt-get update

 Install necessary packages for Docker installation:

sudo apt-get install -y apt-transport-https ca-certificates curl software-properties-common

 Download and add Docker's official GPG key:

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

 Set up the Docker repository:

echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

 Update the package lists again to include Docker's repository:

sudo apt-get update

 Install Docker packages(Docker Engine, CLI, and containerd):

sudo apt-get install -y docker-ce docker-ce-cli containerd.io

 Check Docker service status:

sudo systemctl status docker

 Add the current user to the Docker group for non-root Docker usage:

sudo usermod -aG docker $USER

 Run the hello-world Docker container to verify the installation:

docker run hello-world

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

1. In this step, we installed and set up Minikube and Knative on Ubuntu in WSL2. We downloaded and installed Minikube and Kubectl, set up Knative Serving and Eventing, and checked to make sure everything was working.

 Download Minikube:

curl -LO <https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64>

 Installing Minikube

sudo install minikube-linux-amd64 /usr/local/bin/minikube

 Start Minikube with Docker driver:

minikube start --driver=docker

 Download Kubectl.:

curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"

 Install kubectl:

sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

 Apply Knative Serving CRDs and core components:

kubectl apply --filename https://github.com/knative/serving/releases/download/knative-v1.14.1/serving-crds.yaml

kubectl apply --filename <https://github.com/knative/serving/releases/download/knative-v1.14.1/serving-core.yaml>

 Configure Knative Serving with a default domain:

kubectl apply -f <https://github.com/knative/serving/releases/download/knative-v1.14.1/serving-default-domain.yaml>

 Apply Knative Net-Istio components:

kubectl apply -f https://github.com/knative/eventing/releases/download/knative-v1.14.2/eventing-crds.yaml

kubectl apply -f <https://github.com/knative/eventing/releases/download/knative-v1.14.2/eventing-core.yaml>

* Verifying Knative Setup

kubectl get pods -n knative-serving

kubectl get pods -n knative-eventing

kubectl get services -n istio-system

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. In this step, we installed and set up Homebrew on WSL2. Then, we installed the Knative client (kn) and Kind. We used the Knative Quickstart plugin to set up Knative on the local Kubernetes cluster managed by Kind. Finally, we deployed a sample Knative service to make sure everything was working.

 Starting Minikube:

minikube start

 Download and install Homebrew, then add Homebrew to the system PATH:

/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

echo 'eval "$(/home/linuxbrew/.linuxbrew/bin/brew shellenv)"' >> /home/$(whoami)/.profile

eval "$(/home/linuxbrew/.linuxbrew/bin/brew shellenv)"

 Install the Knative client (kn) using Homebrew.

brew install knative/client/kn

 Download and install Kind (Kubernetes in Docker) by moving it to the local bin directory:

curl -Lo ./kind https://kind.sigs.k8s.io/dl/latest/kind-linux-amd64

chmod +x ./kind

sudo mv ./kind /usr/local/bin/kind

 Install the Knative Quickstart plugin using Homebrew and then use it to set up a Knative environment with Kind:

brew install knative-extensions/kn-plugins/quickstart

kn quickstart kind

 Create and deploy a Knative service named hello using the specified Docker image and environment variable:

kn service create hello \

--image ghcr.io/knative/helloworld-go:latest \

--port 8080 \

--env TARGET=World

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated



The deployment resulted in a successfully running Knative service. The service was accessible at the URL: http://hello.default.127.0.0.1.sslip.io

1. After ensuring that the knative works properly its time to create our function

the project directory contains the following files:

* Dockerfile
* app.py
* service.yaml
* requirements.txt
* input directory
* output directory

A screenshot of a computer

Description automatically generated

Dockerfile

This Dockerfile sets up the environment to run Flask application with all the necessary dependencies.

A computer screen shot of a black screen

Description automatically generated

requirements.txt

This file include the Python dependencies required for your Flask application.

A screenshot of a computer

Description automatically generated

app.py

This Python file uses Flask to create a web application that allows users to upload a video and convert it to images using ffmpeg

A screenshot of a computer

Description automatically generated

service.yaml

This Kubernetes service configuration file deploys the Docker container in a Kubernetes cluster using Knative.

A screenshot of a computer

Description automatically generated

Input and Output Directories

The input directory is for storing uploaded videos, and the output directory is for storing the extracted images from the videos.

1. Commands to Build and Run the Project:

 Starting Minikube:

minikube start

 Navigating to the Project Directory:

cd video-to-image

 Building the Docker Image:

docker build -t aryakh22/video-to-image:latest .

This command creates a Docker image with the tag aryakh22/video-to-image:latest.

 Pushing the Docker Image to Docker Hub:

docker push aryakh22/video-to-image:latest

 Applying the Kubernetes Service Configuration:

kubectl apply -f service.yaml

 Checking Kubernetes Deployments:

kubectl get deployments

to see if video-to-image is available and ready.

 Checking Kubernetes Pods:

kubectl get pods

This lists all the pods running in the cluster. The pods associated with your service are listed and shown as running.

Checking Knative Service:

kubectl get ksvc video-to-image

This lists the Knative services. Your service video-to-image is listed, indicating it's managed by Knative.

 Port Forwarding to Access the Service:

kubectl port-forward pod/video-to-image-00001-deployment-7988f5cc6f-pd9kc 8080:8080

This forwards port 80 of the video-to-image-00001-deployment-7988f5cc6f-pd9kc pod to port 8080 on your local machine. This allows you to access the service running inside the Kubernetes cluster from your local machine by going to http://localhost:8080.

A screenshot of a computer

Description automatically generated

### Repetitive step of the deployment 1. Start Minikube

minikube start

2. Navigate to the Project Directory

cd video-to-image

3. View Deployments

kubectl get deployments

* Initially there are no deployments listed.

4. View Pods

kubectl get pods

* Initially, there are no pods listed.

5. View Knative Service

kubectl get ksvc video-to-image

* Shows the URL, latest revision, readiness status, and any reasons for unavailability.

6. Delete Knative Service

kubectl delete ksvc video-to-image

* Deletes the Knative service video-to-image.
* Necessary when we need to reapply the service with new changes or configurations.
* We need to do this step everytime

7. Apply the Service YAML

kubectl apply -f service.yaml

* Applies the configuration specified in service.yaml Creates a new deployment and Knative service.

8. View Updated Deployments

kubectl get deployments

* Lists the new deployments created.
* Shows the deployment name, desired replicas, current state, and age.

9. View Updated Pods

kubectl get pods

* Lists the new pods created by the deployment.
* Shows the pod name, ready status, restarts, and age.

10. View Updated Knative Service

kubectl get ksvc video-to-image

* Lists the updated status of the specified Knative service.
* Confirms the readiness of the new revision.

11. Port Forward to Access Service

kubectl port-forward pod/video-to-image-00001-deployment-<pod-id> 8080:8080

* Forwards the local port 8080 to port 80 of the specified pod.
* Allows accessing the service running in the pod locally via localhost:8080

A screenshot of a computer screen

Description automatically generated

We have two ways to run the "Video to Image Converter" application:

1. Using a Web Browser:
   * Access the application through a web interface by navigating to http://localhost:8080 in browser. This allows to upload a video file and convert it to images using the graphical interface.
   * A computer screen with a white screen

     Description automatically generated
2. Using a Curl Command:
   * we can interact with the application via the command line using a curl command. This method involves sending a POST request to the server to upload the video and trigger the conversion process.
   * **curl -X POST http://localhost:8080/convert -F "video=@input/Video.mp4"**

A screenshot of a computer screen

Description automatically generated

Here we can see the input folder containing the input video and output folder containing the frames

A screenshot of a computer

Description automatically generated

After some minutes we encounter an error which means that the connection to the pod was lost or that the container inside the pod was not found. This situation requires repeating the deployment steps to ensure the service is correctly up and running.

A screenshot of a computer screen

Description automatically generated

Then we install EOG tool to open the frames

A computer screen shot of a computer screen

Description automatically generated

References

1. [Knative Documentation](https://knative.dev/docs/)
2. [Kubernetes Documentation](https://kubernetes.io/docs/home/)
3. [Minikube Documentation](https://minikube.sigs.k8s.io/docs/start/?arch=%2Fwindows%2Fx86-64%2Fstable%2F.exe+download)
4. [Docker Documentation](https://docs.docker.com/" \t "_new)
5. [Flask Documentation](https://flask.palletsprojects.com/en/3.0.x/)
6. [FFmpeg Documentation](https://ffmpeg.org/documentation.html" \t "_new)
7. [MoviePy Documentation](https://moviepy.readthedocs.io/en/latest/index.html)
8. [EOG Documentation](https://help.gnome.org/users/eog/stable/)