6 - Deploy Video Processing Service

10:25

14:28

Mark Lesson Complete

Full Stack Development

19 / 22

Intro

0 Demo and Architecture 8 min FREE

1 F 4 min FREE

Video Processing Service

Initialize

Host Video Processing Service on Google Cloud Run

We will now deploy our service to Google **Cloud Run**, which is a fully-managed serverless container platform. That means we don't need to worry about managing servers or scaling our service.

Video Initialize

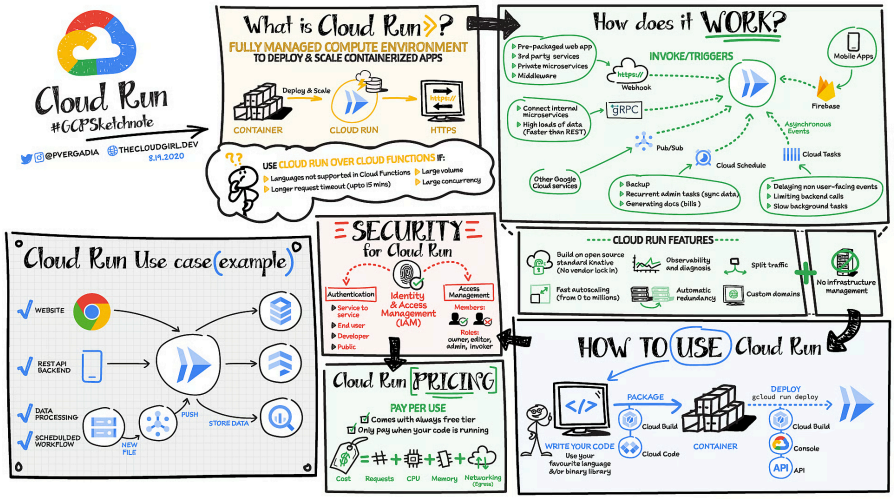
2 Video Processing Service 11 min FREE

3 Process Video Locally 13 min FREE

4 Containerize Processing Service 15 min

5 Convert Videos Hosted on Google Cloud Storage 24 min

Google Cloud



There are a few steps involved:

1. Create a Google Cloud Project and enable billing (in our case, we will create a Firebase project).
2. Install `gcloud` and `gsutil` CLI tools
3. Upload our Docker image to Google Artifact Registry
4. Deploy our Docker image to Google Cloud Run

1. Create a Firebase Project

We will create a Firebase project, which will also create a Google Cloud project for us.

You can do this via the Firebase console:

https://console.firebase.google.com/

Take note of the `project id` since we will need it throughout the course.

This will create a Google Cloud project for us with the same `project id`, which we can see in the Google Cloud console:

https://console.cloud.google.com/

> Google Cloud offers a free tier, as well as $300 of free credit. Which should be more than enough for this course.
> **https://cloud.google.com/free/docs/free-cloud-features**

2. Install `gcloud` and `gsutil` CLI tools

Install `gcloud` and `gsutil` CLI tools. Then initialize `gcloud` and authenticate with your project.

The steps will depend on your operating system, so it's best to refer to the official docs.

https://cloud.google.com/sdk/docs/install

Make sure to authentication with your account and set your project:

```
gcloud auth login # Copy the output url and paste it into your browser

gcloud config set project <PROJECT_ID>
```

3. Upload our Docker image to Google Artifact Registry

Enable artifact registry

```
gcloud services enable
artifactregistry.googleapis.com
```

(Optional) Update gcloud components

```
gcloud components update
```

Create an Artifact Registry repository:

```
gcloud artifacts repositories create video-
processing-repo \
  --repository-format=docker \
  --location=us-central1 \
  --description="Docker repository for video
processing service"
```

> For some reason I had an issue with this command, so I created the repo via the console: **https://console.cloud.google.com/artifacts**

Then, rebuild your Docker image. With this naming scheme, docker knows where to push the image and which project.

```
docker build -t us-central1-
docker.pkg.dev/<PROJECT_ID>/video-processing-
repo/video-processing-service .
```

> Important: If you are using mac, add `--platform linux/amd64` to the above command.

You may still need to configure Docker to use gcloud as the credential helper:

```
gcloud auth configure-docker us-central1-
docker.pkg.dev
```

Then, push the Docker image to Google Artifact Registry:

```
docker push us-central1-
docker.pkg.dev/<PROJECT_ID>/video-processing-
repo/video-processing-service
```

4. Deploy our Docker image to Google Cloud Run

Lastly, deploy to Cloud Run.

```
# Enable cloud run
gcloud services enable run.googleapis.com

# Deploy container to cloud run
gcloud run deploy video-processing-service --
image us-central1-
docker.pkg.dev/PROJECT_ID/video-processing-
repo/video-processing-service \
  --region=us-central1 \
  --platform managed \
  --timeout=3600 \
  --memory=2Gi \
  --cpu=1 \
  --min-instances=0 \
  --max-instances=1 \
  --ingress=internal
```

We are setting the ingress to internal so that only GCP internal services can access it. This is so that our Pub/Sub service can invoke it, but not the outside world.

In addition, it might be good to use the `--no-allow-unauthenticated` flag to limit other internal services. Meaning we can set it up so that only our Pub/Sub service can invoke it and not our other GCP internal services. But this would be cumbersome to setup, so we will skip it for now.