20 - Deploy Web App

09:47

Deploy Web Client to Cloud Run

Before we deploy, there's a smaller error we have to fix. (Currently we will get a error when we try to run `npm run build` in the `yt-web-client`).

When you render multiple components, like we do when we list the videos, you need to add a `key` prop to each component. This is a React requirement.

Update your `Link` element in `page.tsx` to include a `key` prop:

```
<Link href={`/watch?v=${video.filename}`}
key={video.id}>
  <Image src={'/thumbnail.png'} alt='video'
width={120} height={80}
    className={styles.thumbnail}/>
</Link>
```

After that, we also have to correct the default caching behaviour of Next.js 13. Which is automatically set to always cache every server-side Fetch request.

If we don't do this, we will not be able to see new videos rendered on the home page, after we upload them.

> Why doesn't this happen in development mode?
> Because in development mode, Next.js will not cache any server-side Fetch requests.
> This has been a source of confusion for many developers, so I hope this clears it up.

We can correct this by adding this line of code to the bottom of our `page.tsx` file:

```
export const revalidate = 30;
```

It will rerun the `getVideos()` function every 30 seconds and update the page with the new videos. This is fine since the API call to `getVideos` will be the same for every user anyway.

> Since we can't specify the caching behaviour within the firebase function call, we use segment level caching. Which sets the caching behaviour for the entire page.

Source: **Next.js Segment-level Caching**

1. Create a Dockerfile

Create a Dockerfile which will be used to build our Docker image:

```
# Stage 1: Build stage
FROM node:18 AS builder

# Set the working directory
WORKDIR /app

# Copy package*.json files
COPY package*.json ./

# Install all dependencies
RUN npm install

# Copy other source code files
COPY . .

# Build the app
RUN npm run build

# Stage 2: Production stage
FROM node:18

# Set the working directory
WORKDIR /app

# Copy package*.json files
COPY package*.json ./
```

```
# Install only production dependencies
RUN npm install --only=production

# Copy built app from the builder stage
COPY --from=builder /app/.next ./.next
COPY --from=builder /app/public ./public

# Expose the listening port
EXPOSE 3000

# Run the app
CMD ["npm", "start"]
```

2. Build and push Docker image to Google Artifact Registry

Create another repo in Google Artifact Registry, this time for the `yt-web-client`:

```
gcloud artifacts repositories create yt-web-client-repo \
  --repository-format=docker \
  --location=us-central1 \
  --description="Docker repository for youtube web client"
```

Alternatively, you could reuse the `video-processing-repo` repo we created in the previous section.

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

```
docker build -t us-central1-docker.pkg.dev/<PROJECT_ID>/yt-web-client-
```

```
repo/yt-web-client .
```

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

Then, push the Docker image to Google Artifact Registry:

```
docker push us-central1-docker.pkg.dev/<PROJECT_ID>/yt-web-client-repo/yt-web-client
```

3. Deploy Docker image to Cloud Run

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

This time we don't need to specify the ingress as internal only.

4. Test the web client

Visit the URL provided by Cloud Run to test the web

Mark Lesson Complete

because we haven't configured the OAuth consent screen yet.

Copy the URL and navigate to Firebase Auth in the console. Go to the `Settings` tab and scroll down to `Authorized domains`. Add the URL to the list of authorized domains.

Now you should be able to sign in and out, as well as upload videos. After uploading a video, you should see it listed on the home page and be able to watch it (after it's completed processing).