Matthias Eder, 11378093
Eric Zhang, n10319191

CAB432 Cloud Computing
Assignment 2 Proposal

QUT, Semester 2
16-10-2022

PicPro

1. Application Description

With our proposed application, users can upload images which are then transcoded and transformed according to a specified combination of various predefined options. The application supports uploading and transcoding the following formats: JPEG, PNG, WebP, GIF, AVIF, TIFF and SVG. A further possible transformation is resizing the image to a user-defined width and height. An image can be rotated, resized and the resolution can be changed.
When a user uploads an image, it will be analysed and its metadata will be displayed to the user. In addition to uploading a new image for transformation, users can also choose from previously uploaded images which will then be available as input for further transformation. The user can additionally save a combination of transformation options as presets. The presets will be displayed to the user as a list and can be loaded into the application. This automatically fills the transformation options with the preset values and the user can apply this transformation to the currently chosen image.
The application will act as a global image transformation platform without a user management. Thus, all uploaded images will be available to all users. This removes the need of limiting specific images only to specific users.
Image transcoding and transformation will be achieved by using the Node.js image processing library `sharp`. As this module will be integrated into the backend and executed there, we do not offload computation to an external API. Load will be generated by increasing the CPU usage due to the processing of large images and increasing the number of processing requests to the backend. This should easily be achievable using Postman or cURL.

2. Architecture and Implementation Phases

The application will be split into a frontend and a backend project. The frontend will be a simple SPA built with Angular and is responsible for displaying the results of the server requests. The other main task of the frontend is the processing of user input like the transformation options for an image. The backend is a simple `Node.js` backend using the web application framework `express.js`. It is responsible for image processing/transformation, storing and retrieving images and presets from persistence services, and analysing images with regard to their metadata. The backend will be subject to automatic scaling. The application does not rely on any form of state and can thus be described as stateless.

The implementation phases are split in the following way:

1. Brainstorming for scenario and planning the architecture
2. Writing the proposal
3. Setup of frontend and backend projects as well as cloud services for persistence
4. Developing first prototype with focus on image transformation endpoint on backend
5. Testing prototype with cloud infrastructure to validate scaling
6. Implement remaining features for frontend (Matthias) and backend (Eric)

7. Dockerize application, deploy on AWS
8. Test, refine, validate

Figure 1 shows the proposed architecture of our application with regard to the cloud services used to host the application, provide persistence services and to balance the load between instances.

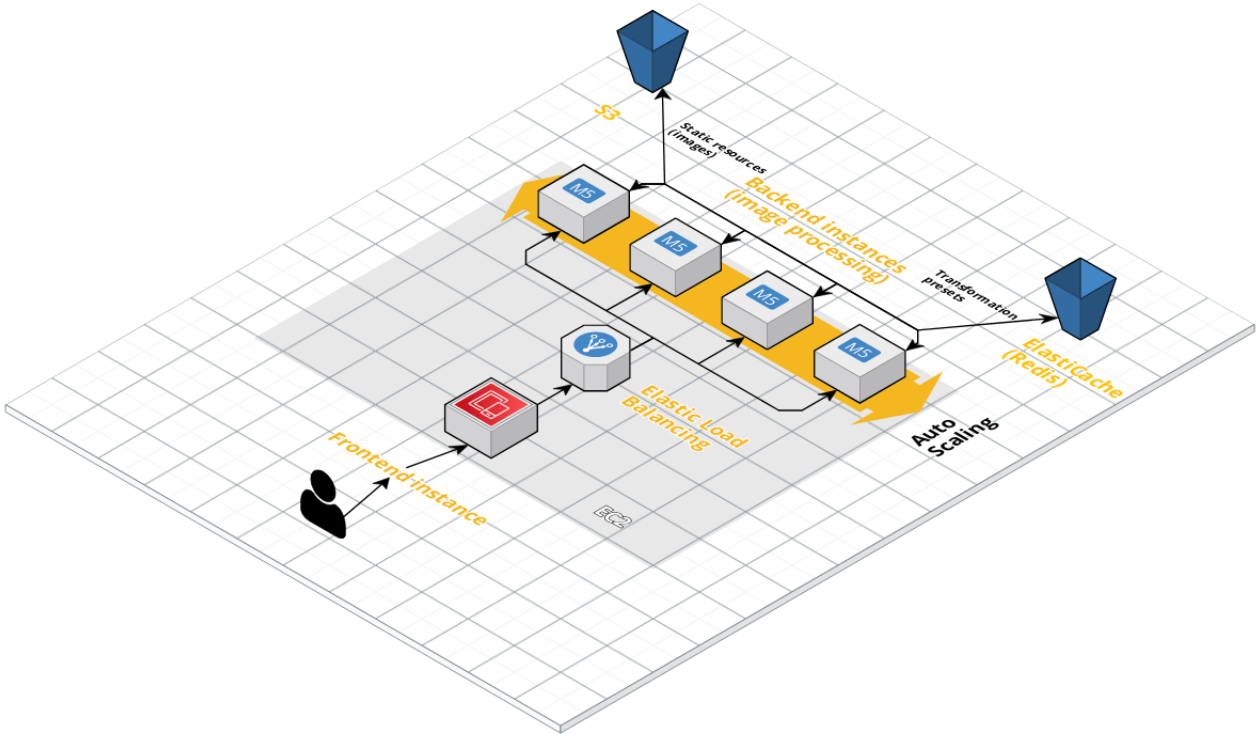

Figure 1: Proposed application architecture

3. Persistence, Scaling and Application Thresholds

We will be using two distinct persistence services: AWS S3 and AWS ElastiCache for Redis. S3 will be used to store input and output images before and after pre-processing. Redis will be used as a caching service to store the transformation presets that users can create.
The only relevant application threshold is the CPU usage which will serve as the metric used for scaling. The exact value is hard to predict without prior testing but will most likely be around 50% to 70% to allow for short-term spikes in the usage which might occur naturally during the processing of single computationally expensive transformation while also keeping costs low as new instances will only be created when it is strictly necessary.

4. Example Use Cases

There are many use cases for this application. One would be to rescale images. A user can import an image, select a new size, and have the application output the rescaled image. Another use case would be to change the file type. A user can transcode an imported image to a different output. (JPEG, PNG, WebP, AVIF, TIFF, GIF). Additionally, filters could be applied to the image. A user can use an input image and modify it to be grey scale, black white, blurred or change the hue/saturation.

5. API Description

The API that will be used is a Node.js module called `sharp`. `sharp` allows you to modify an image in a variety of ways. A user can trans code an image (PNG -> JPEG), crop an image, add a colour filter like grey scale or black white, change hue or saturation, blur the image, and rotate it as well.