Markdown Converter Infrastructure Design

Cloud Run Services

The frontend and backend API is handled by Cloud Run Services. Each container serves the frontend using a basic python Flask application, and handles `/api/` requests that come in from the client.

API Endpoints:

- `/api/mdconvert` handles POST requests containing markdown from the client that needs to be converted and creates a pubsub message with the final GCS bucket that the converted HTML should be placed in. Once we publish a message to pubsub, we then return the `/api/retrieve/<filename>` endpoint that the client should poll against until the file is done. The filename will be the pubsub message ID appended to `.html`.
- `/api/retrieve/<filename>` handles GET requests from the client waiting for a converted file. Returns 404 if the file is not located on the GCS mountpoint. This is expected until the file is converted and uploaded to the GCS bucket.
- `/health` health check endpoint for Cloud Run to determine container status.

GCS Mounts:

- Each tenant (aka customer) has a Cloud Run Service per environment (stage, prod). They also have a GCS bucket that their converted files will be uploaded to and retrieved from. Each Cloud Run Service will only have IAM access to their appropriate bucket and have that mounted at `/mnt/bucket` for the API to easily retrieve converted files.

Pub/Sub

Each environment (stage, prod) has a pub/sub topic. When messages are published they will be consumed by the appropriate Cloud Function for the environment.

Cloud Functions

Each environment (stage, prod) has a Cloud Function. This function’s purpose is to take messages from the appropriate Pub/Sub topic and convert the markdown in the message data into html. Then the output html is uploaded to the appropriate GCS bucket defined by the message attribute `bucketname` using the message ID in the format `<messageId>.html`

GCS

Each tenant (aka customer) has a GCS bucket for each environment (stage, prod) that their converted html files will be stored in.