Notes

Cloud Functions

See the Cloud Functions docs for Firebase.

Functions samples

See the official GitHub repo of Cloud Functions for Firebase sample functions

Mount an Express app

```
const functions = require('firebase-functions');
const express = require('express');
const cors = require('cors');
const app = express();

// Automatically allow cross-origin requests
app.use(cors({ origin: true }));

// Add middleware to authenticate requests
app.use(myMiddleware);

// build multiple CRUD interfaces:
app.get('/:id', (req, res) => res.send(Widgets.getById(req.params.id)));
app.post('/', (req, res) => res.send(Widgets.create()));
app.put('/:id', (req, res) => res.send(Widgets.update(req.params.id, req.body)));
app.delete('/:id', (req, res) => res.send(Widgets.delete(req.params.id)));
app.get('/', (req, res) => res.send(Widgets.list()));

// Expose Express API as a single Cloud Function:
exports.widgets = functions.https.onRequest(app);
```

Mount an Express handler

```
exports.helloWorld = functions.https.onRequest((req, res) => {
  res.status(200);
  res.send('hello world');
});
```

Firestore triggers

- onCreate
- onUpdate
- onDelete
- onWrite

```
exports.createUser = functions.firestore.document('users/{userId}').onCreate(event
=> {
  // Get an object representing the document
  // e.g. {'name': 'Marie', 'age': 66}
  var newValue = event.data.data();

  // access a particular field as you would any JS property
  var name = newValue.name;

  // perform desired operations ...
});
```

Realtime Database triggers

- onCreate
- onUpdate
- onDelete
- onWrite

```
exports.makeUppercase =
functions.database.ref('/messages/{pushId}/original').onWrite(event => {
  // Grab the current value of what was written to the Realtime Database.
  const original = event.data.val();
  console.log('Uppercasing', event.params.pushId, original);
  const uppercase = original.toUpperCase();
  // You must return a Promise when performing asynchronous tasks inside a
Functions such as
  // writing to the Firebase Realtime Database.
  // Setting an "uppercase" sibling in the Realtime Database returns a Promise.
  return event.data.ref.parent.child('uppercase').set(uppercase);
});
```

Firebase Authentication

- onCreate
- onDelete

```
exports.sendWelcomeEmail = functions.auth.user().onCreate(event => {
  const user = event.data; // The Firebase user.

  const email = user.email; // The email of the user.
  const displayName = user.displayName; // The display name of the user.
});
```

Firebase Storage

- onChange

```
exports.generateThumbnail = functions.storage.object().onChange(event => {
  const object = event.data; // The Storage object.

  const fileBucket = object.bucket; // The Storage bucket that contains the file.
  const filePath = object.name; // File path in the bucket.
  const contentType = object.contentType; // File content type.
  const resourceState = object.resourceState; // The resourceState is 'exists' or
'not_exists' (for file/folder deletions).
  const metageneration = object.metageneration; // Number of times metadata has
been generated. New objects have a value of 1.

  // Exit if this is triggered on a file that is not an image.
  if (!contentType.startsWith('image/')) {
    console.log('This is not an image.');
    return;
  }

  // Get the file name.
  const fileName = path.basename(filePath);
  // Exit if the image is already a thumbnail.
  if (fileName.startsWith('thumb_')) {
    console.log('Already a Thumbnail.');
    return;
  }

  // Exit if this is a move or deletion event.
  if (resourceState === 'not_exists') {
    console.log('This is a deletion event.');
    return;
  }

  // Exit if file exists but is not new and is only being triggered
  // because of a metadata change.
  if (resourceState === 'exists' && metageneration > 1) {
    console.log('This is a metadata change event.');
    return;
  }
});
```

Use ImageMagick

```
const functions = require('firebase-functions');
const gcs = require('@google-cloud/storage')();
const spawn = require('child-process-promise').spawn;
const path = require('path');
const os = require('os');
const fs = require('fs');

exports.generateThumbnail = functions.storage.object().onChange(event => {
  const object = event.data;

  const fileBucket = object.bucket;
  const filePath = object.name;
  const contentType = object.contentType;

  // Download file from bucket.
  const bucket = gcs.bucket(fileBucket);
  const tempFilePath = path.join(os.tmpdir(), fileName);
  const metadata = { contentType: contentType };
  return bucket
    .file(filePath)
    .download({
      destination: tempFilePath,
    })
    .then(() => {
      console.log('Image downloaded locally to', tempFilePath);
      // Generate a thumbnail using ImageMagick.
      return spawn('convert', [tempFilePath, '-thumbnail', '200x200>',
tempFilePath]);
    })
    .then(() => {
      console.log('Thumbnail created at', tempFilePath);
      // We add a 'thumb_' prefix to thumbnails file name. That's where we'll
upload the thumbnail.
      const thumbFileName = `thumb_${fileName}`;
      const thumbFilePath = path.join(path.dirname(filePath), thumbFileName);
      // Uploading the thumbnail.
      return bucket.upload(tempFilePath, { destination: thumbFilePath, metadata:
metadata });
      // Once the thumbnail has been uploaded delete the local file to free up
disk space.
    })
    .then(() => fs.unlinkSync(tempFilePath));
});
```