SWE 432 -Web Application Development

Fall 2022

George Mason University

Dr. Kevin Moran

Week 5: More Microservices!

Administrivia

- ***Midterm Exam*** - Next Thursday, October 6th (will discuss next class)

- ***HW Assignment 2*** - Due October 4th Before Class

    - Accept GitHub Classroom Invitation!!

Class Overview

- Today - *Even More Microservices:* A Few More Concepts and a Demo
    - *In Class Activity:* Building on a Microservice for Jokes (+ HW2 Help)
- Next Class - *Templates, Databinding, and HTML -* Beginning to look at frontend development!

Even More Microservices!

Blobs: Storing uploaded files

- Example: User uploads picture
  - … and then?
  - … somehow process the file?

How do we store our files?

- Dealing with text is easy - we already figured out firebase
  - Could use other databases too… but that’s another class!
- But
  - What about pictures?
  - What about movies?
  - What about big huge text files?
- Aka…Binary Large OBject (BLOB)
  - Collection of binary data stored as a single entity
  - Generic terms for an entity that is array of bytes

Working with Blobs

- Module: multer

- Simplest case: take a file, save it on the server

```
app.post('/upload',upload.single("upload"), function(req, res) {
        var sampleFile = req.file.filename;
      //sampleFile is the name of the file that now is living on our server
        res.send('File uploaded!');
    });
});
```

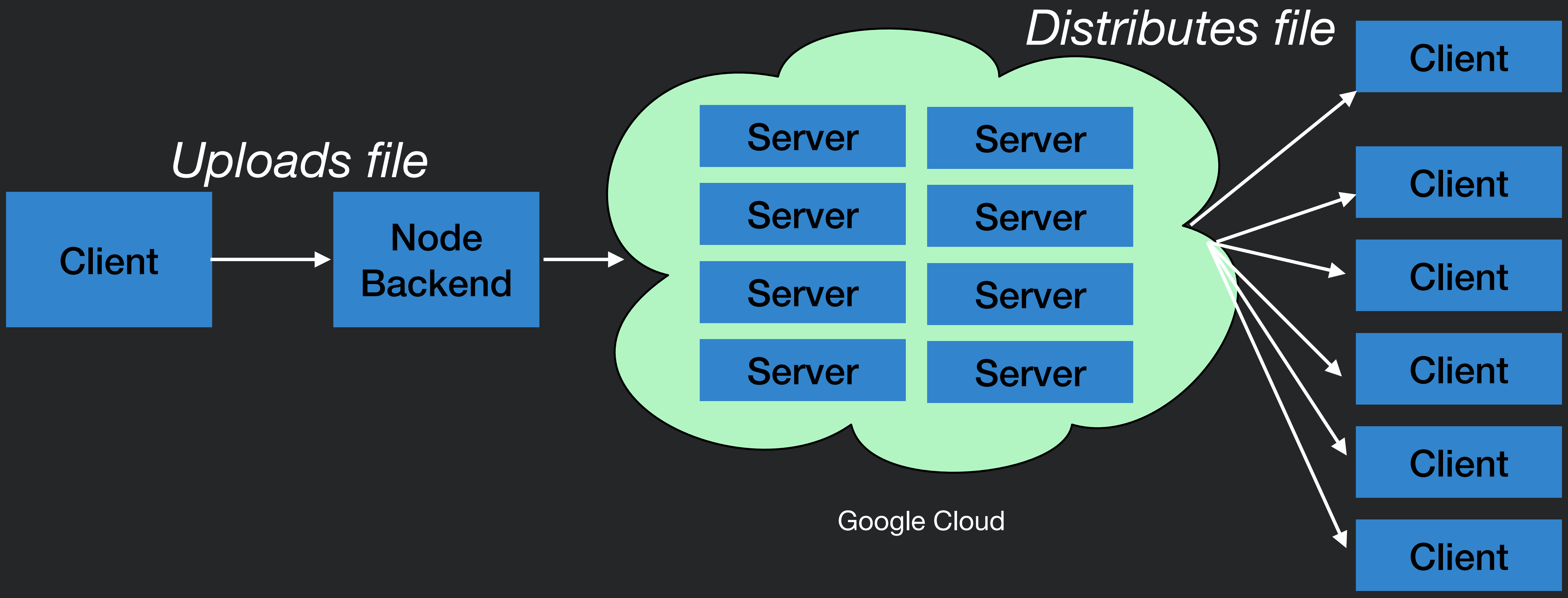
- Long story... can’t easily have file uploads and JSON requests at the same time

Where to store blobs

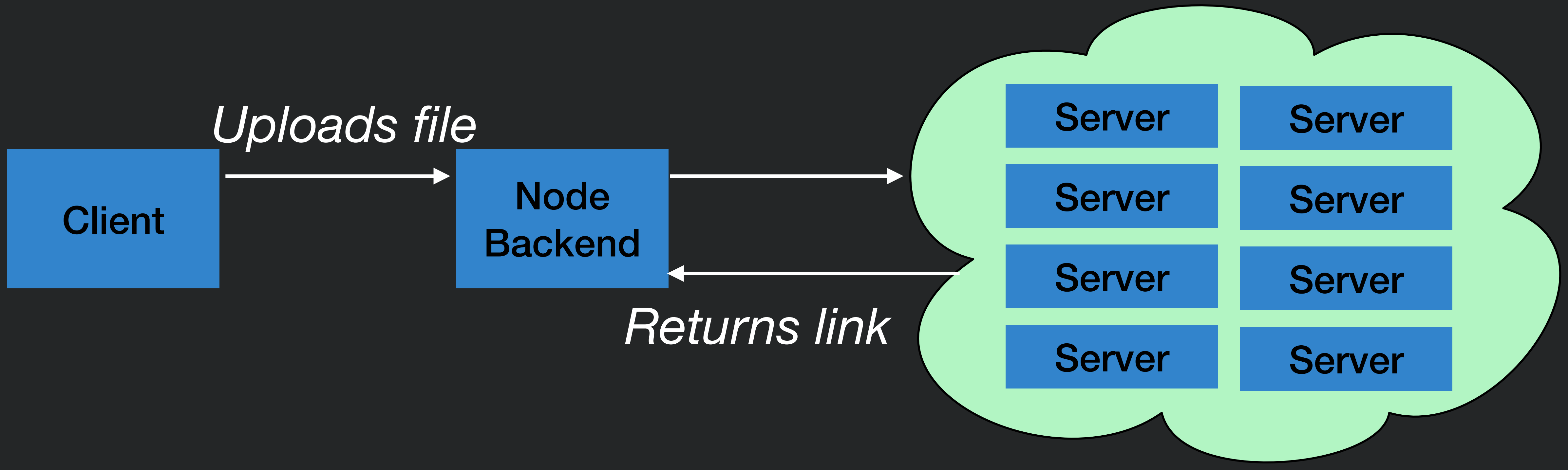
- Saving them on our server is fine, but…
  - What if we don't want to deal with making sure we have enough storage
  - What if we don't want to deal with backing up those files
  - What if our app has too many requests for one server and state needs to be shared between load-balanced servers
  - What if we want someone else to deal with administering a server

Blob stores

- Amazon, Google, and others want to let you use their platform to solve this!



Blob Stores

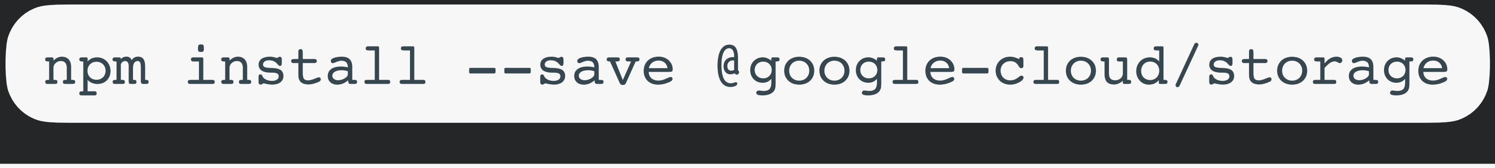


Google Cloud

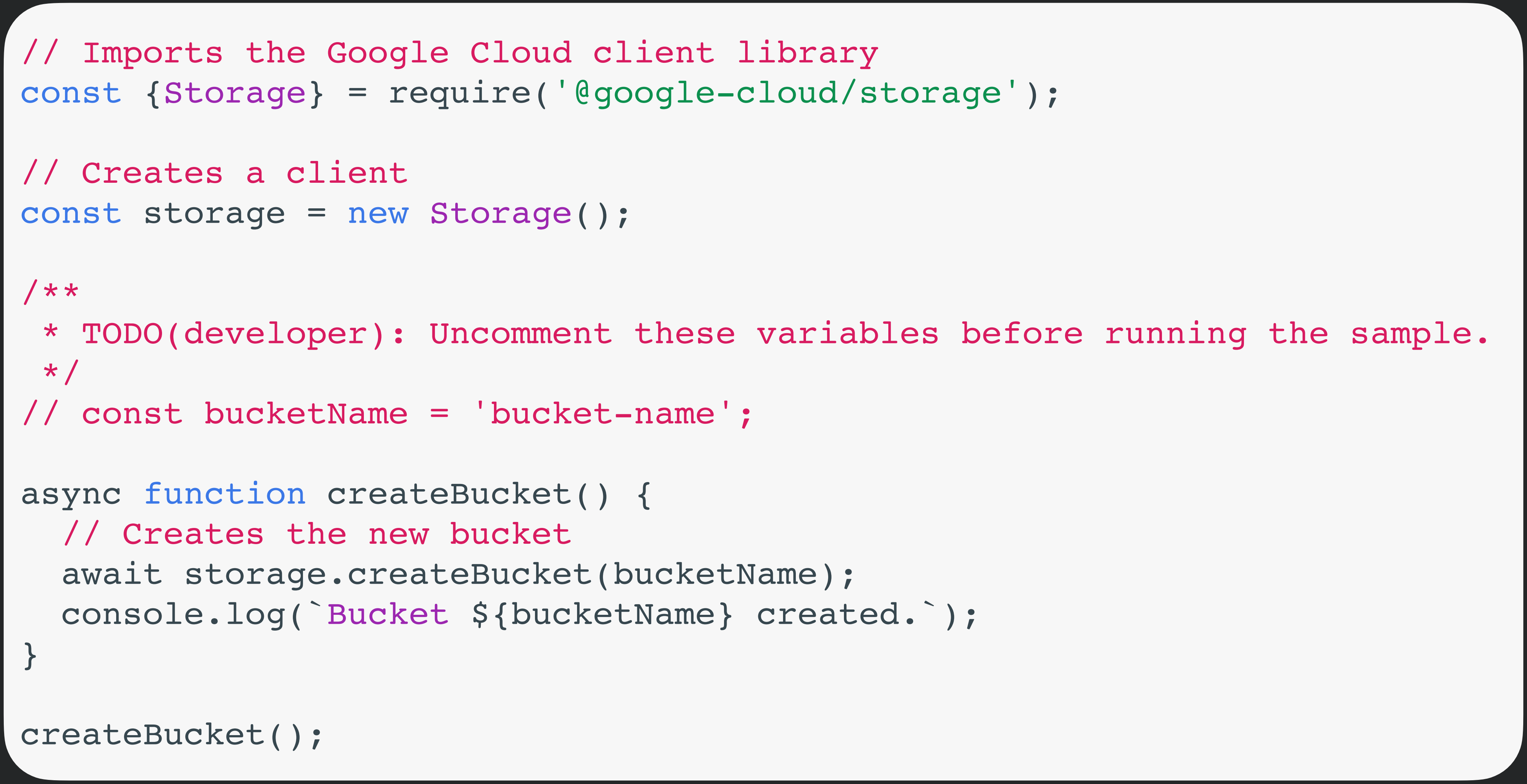
Typical workflow:
Client uploads file to your backend
Backend persists file to blob store
Backend saves link to file, e.g. in Firebase

Google Cloud Storage

- You get to store 5GB for free (but not used in this class)
- Setup



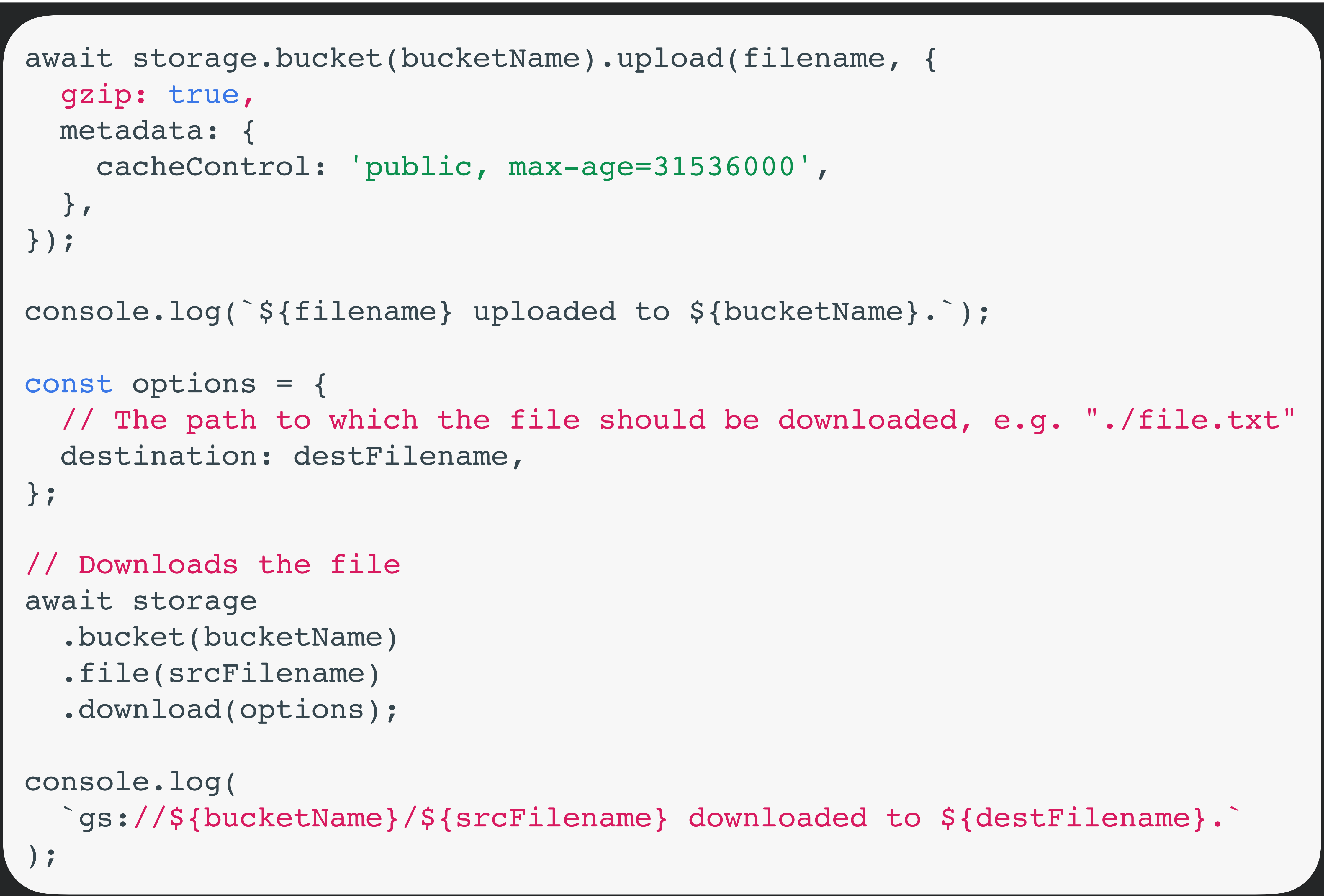
```
npm install --save @google-cloud/storage
```



```
// Imports the Google Cloud client library
const {Storage} = require('@google-cloud/storage');

// Creates a client
const storage = new Storage();

/**
 * TODO(developer): Uncomment these variables before running the sample.
 */
// const bucketName = 'bucket-name';

async function createBucket() {
  // Creates the new bucket
  await storage.createBucket(bucketName);
  console.log(`Bucket ${bucketName} created.`);
}

createBucket();
```

https://cloud.google.com/storage/docs/reference/libraries

Google Cloud Storage



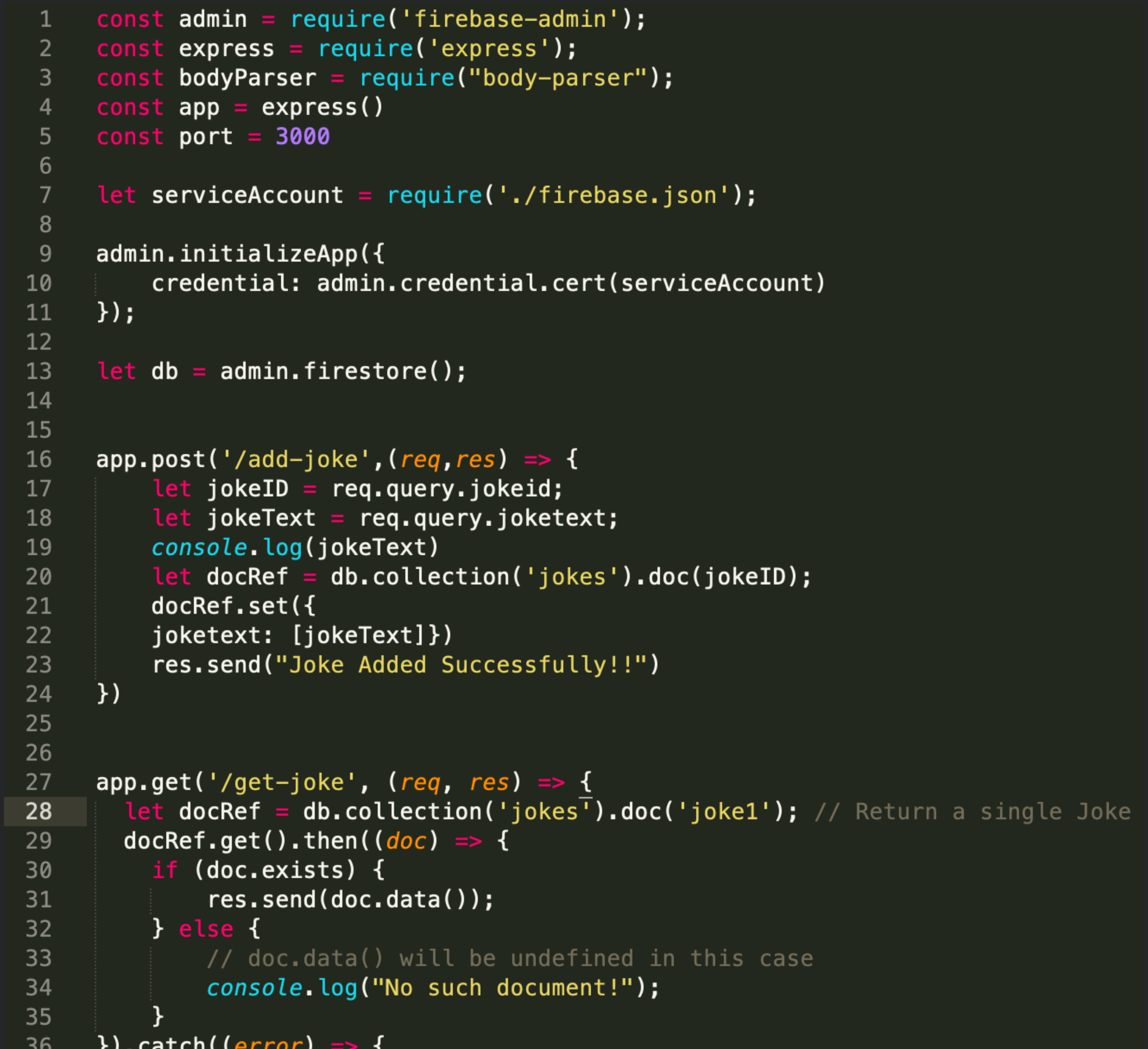
```
await storage.bucket(bucketName).upload(filename, {
  gzip: true,
  metadata: {
    cacheControl: 'public, max-age=31536000',
  },
});

console.log(`${filename} uploaded to ${bucketName}.`);

const options = {
  // The path to which the file should be downloaded, e.g. "./file.txt"
  destination: destFilename,
};

// Downloads the file
await storage
  .bucket(bucketName)
  .file(srcFilename)
  .download(options);

console.log(
  `gs://${bucketName}/${srcFilename} downloaded to ${destFilename}.`
);
```

https://cloud.google.com/storage/docs/reference/libraries

Demo: Let's build a Microservice!

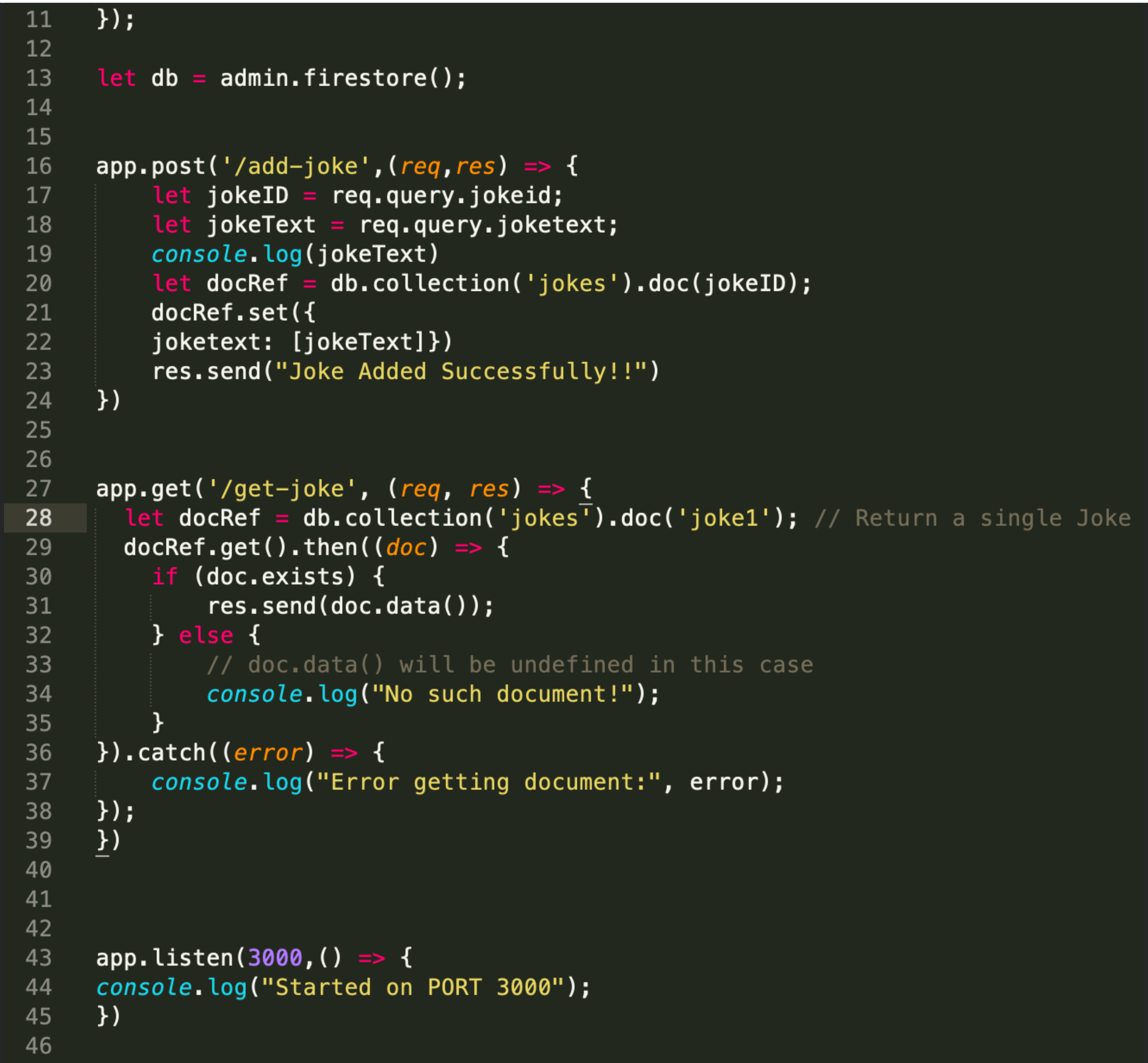
- We've now seen most of the key concepts in building a microservice.
- Let's build a microservice!
    - - Firebase for persistence
    - - Handle post requests
    - Microservice for jokes

Demo: Let's build a Microservice!



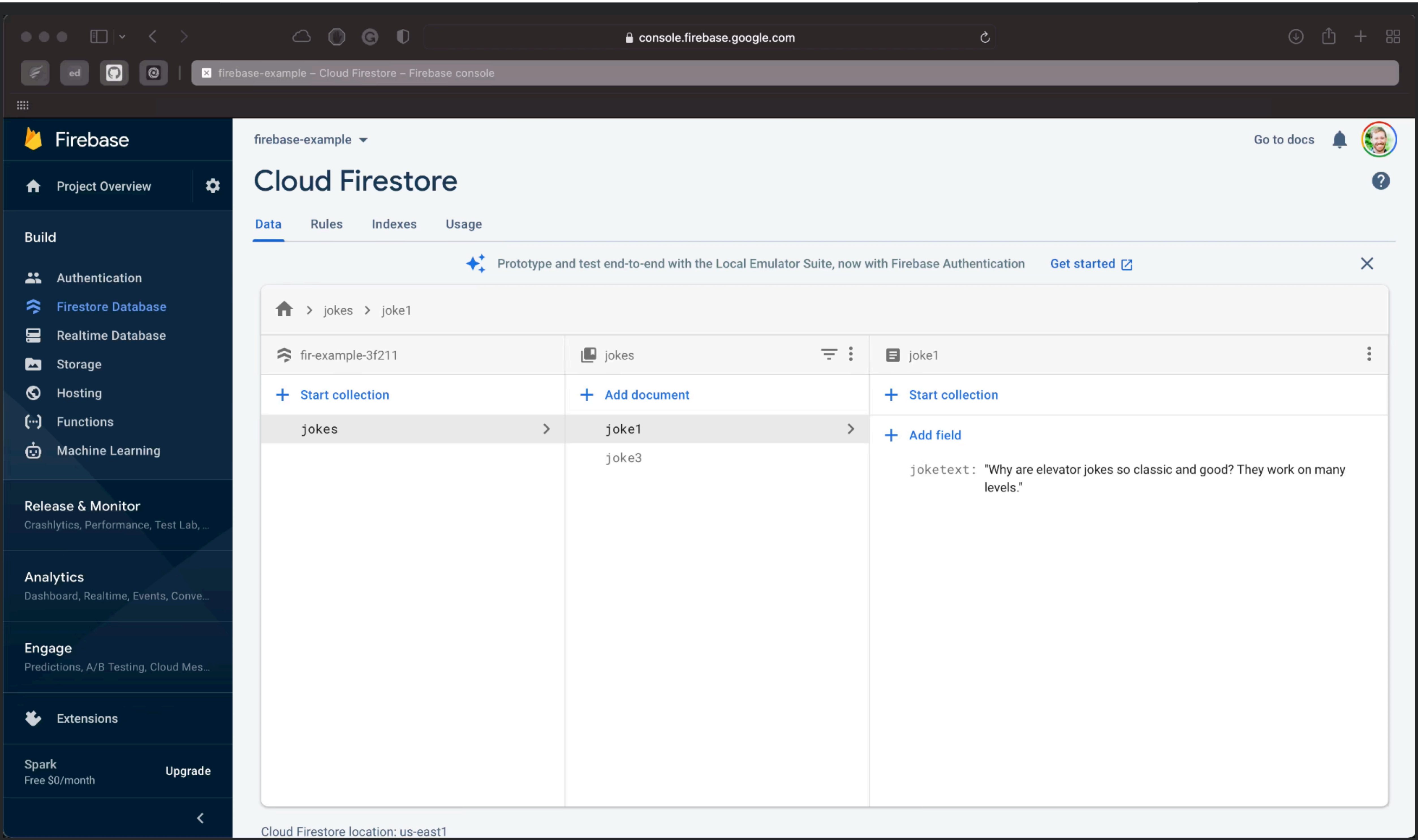
```
const admin = require('firebase-admin');
const express = require('express');
const bodyParser = require("body-parser");
const app = express()
const port = 3000

let serviceAccount = require('./firebase.json');

admin.initializeApp({
    credential: admin.credential.cert(serviceAccount)
});

let db = admin.firestore();


app.post('/add-joke',(req,res) => {
    let jokeID = req.query.jokeid;
    let jokeText = req.query.joketext;
    console.log(jokeText)
    let docRef = db.collection('jokes').doc(jokeID);
    docRef.set({
    joketext: [jokeText]})
    res.send("Joke Added Successfully!!")
})


app.get('/get-joke', (req, res) => {
  let docRef = db.collection('jokes').doc('joke1'); // Return a single Joke
  docRef.get().then((doc) => {
    if (doc.exists) {
        res.send(doc.data());
    } else {
        // doc.data() will be undefined in this case
        console.log("No such document!");
    }
}).catch((error) => {
```

Demo: Let's build a Microservice!



```
});

let db = admin.firestore();


app.post('/add-joke',(req,res) => {
    let jokeID = req.query.jokeid;
    let jokeText = req.query.joketext;
    console.log(jokeText)
    let docRef = db.collection('jokes').doc(jokeID);
    docRef.set({
    joketext: [jokeText]})
    res.send("Joke Added Successfully!!")
})


app.get('/get-joke', (req, res) => {
  let docRef = db.collection('jokes').doc('joke1'); // Return a single Joke
  docRef.get().then((doc) => {
    if (doc.exists) {
        res.send(doc.data());
    } else {
        // doc.data() will be undefined in this case
        console.log("No such document!");
    }
}).catch((error) => {
    console.log("Error getting document:", error);
});
})



app.listen(3000,() => {
console.log("Started on PORT 3000");
})

```

Demo: Let's build a Microservice!



Cloud Firestore

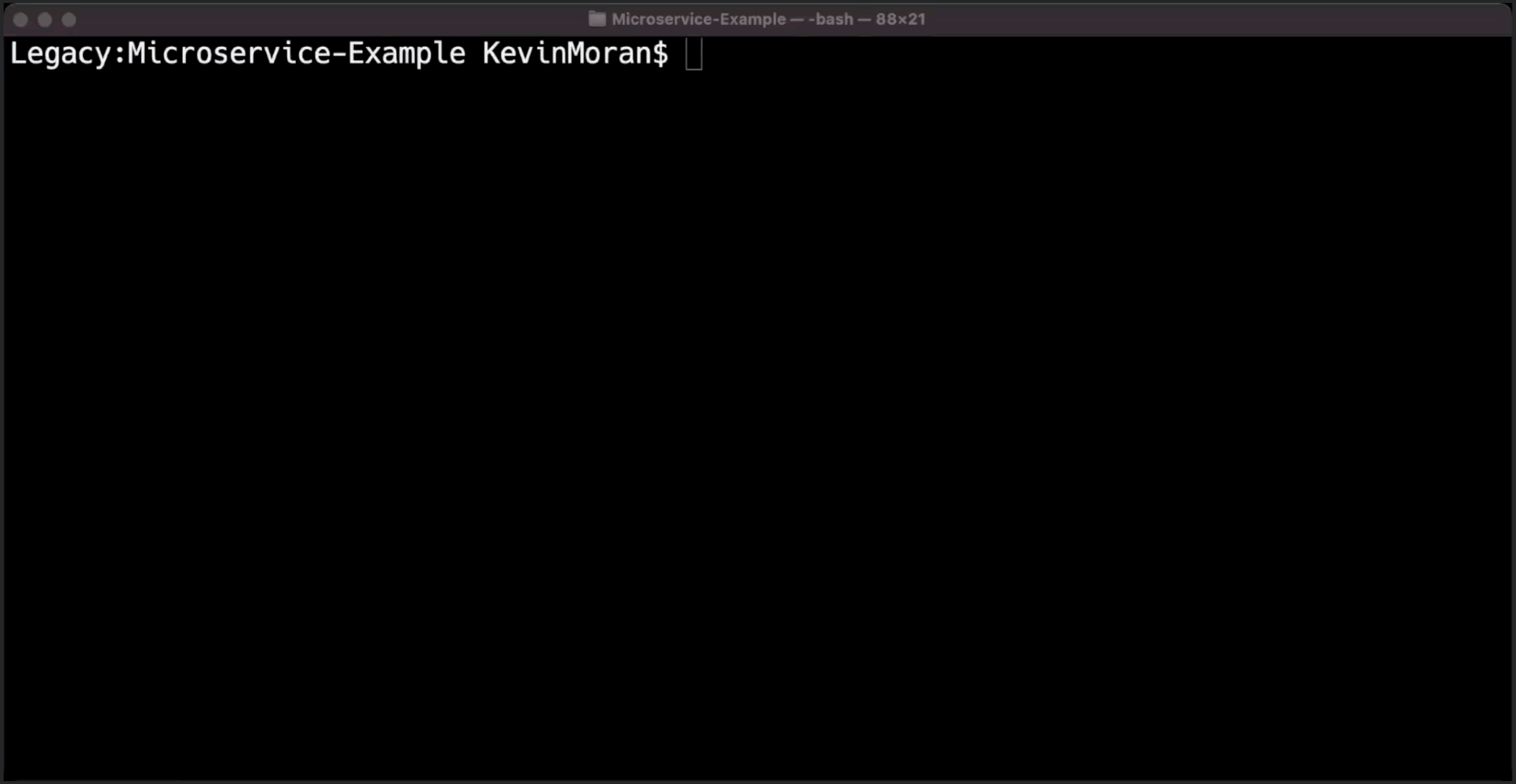
Prototype and test end-to-end with the Local Emulator Suite, now with Firebase Authentication

jokes > joke1

fir-example-3f211	jokes	joke1
jokes	joke1	joketext: "Why are elevator jokes so classic and good? They work on many levels."
	joke3	

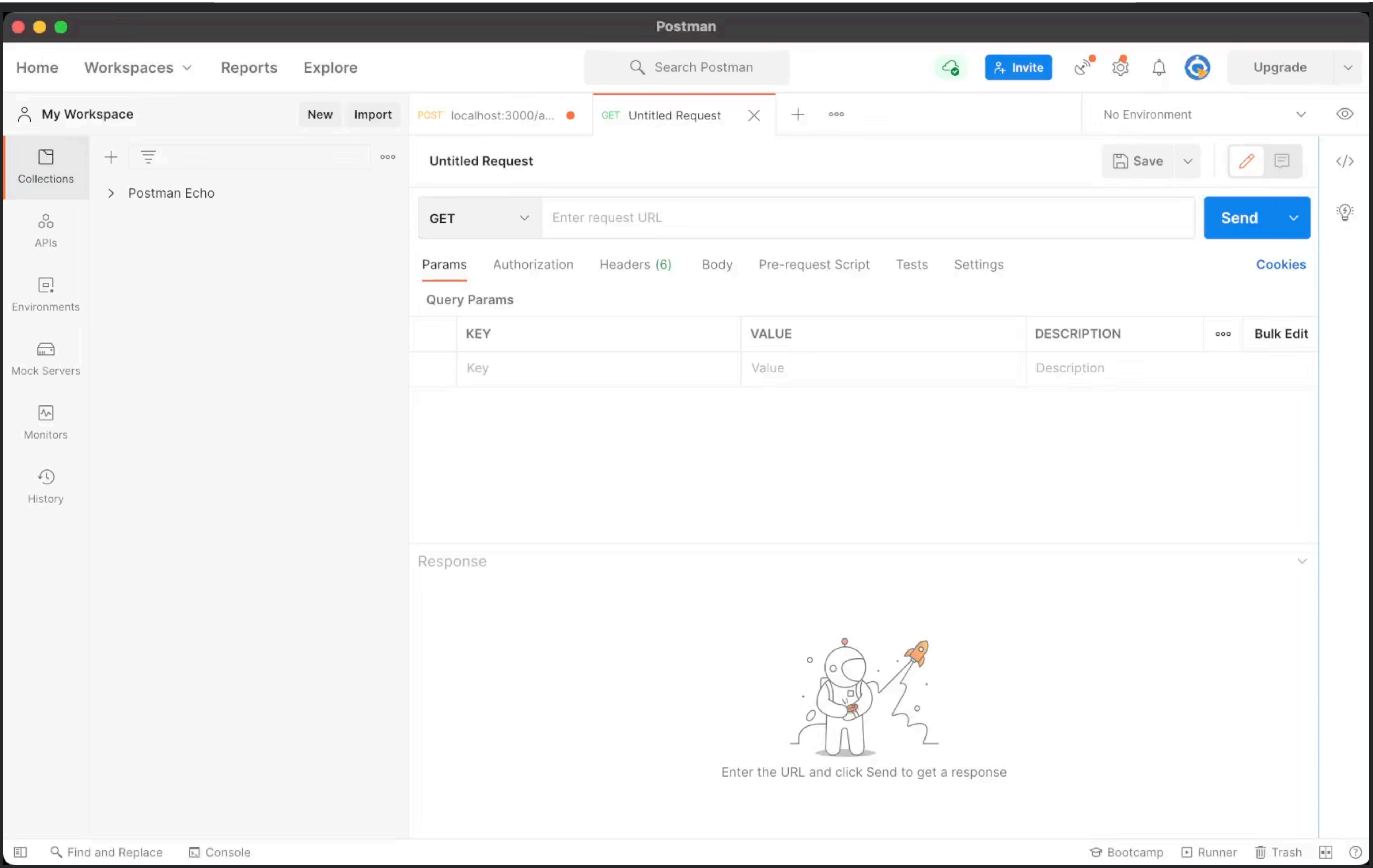
Cloud Firestore location: us-east1

Demo: Let's build a Microservice!

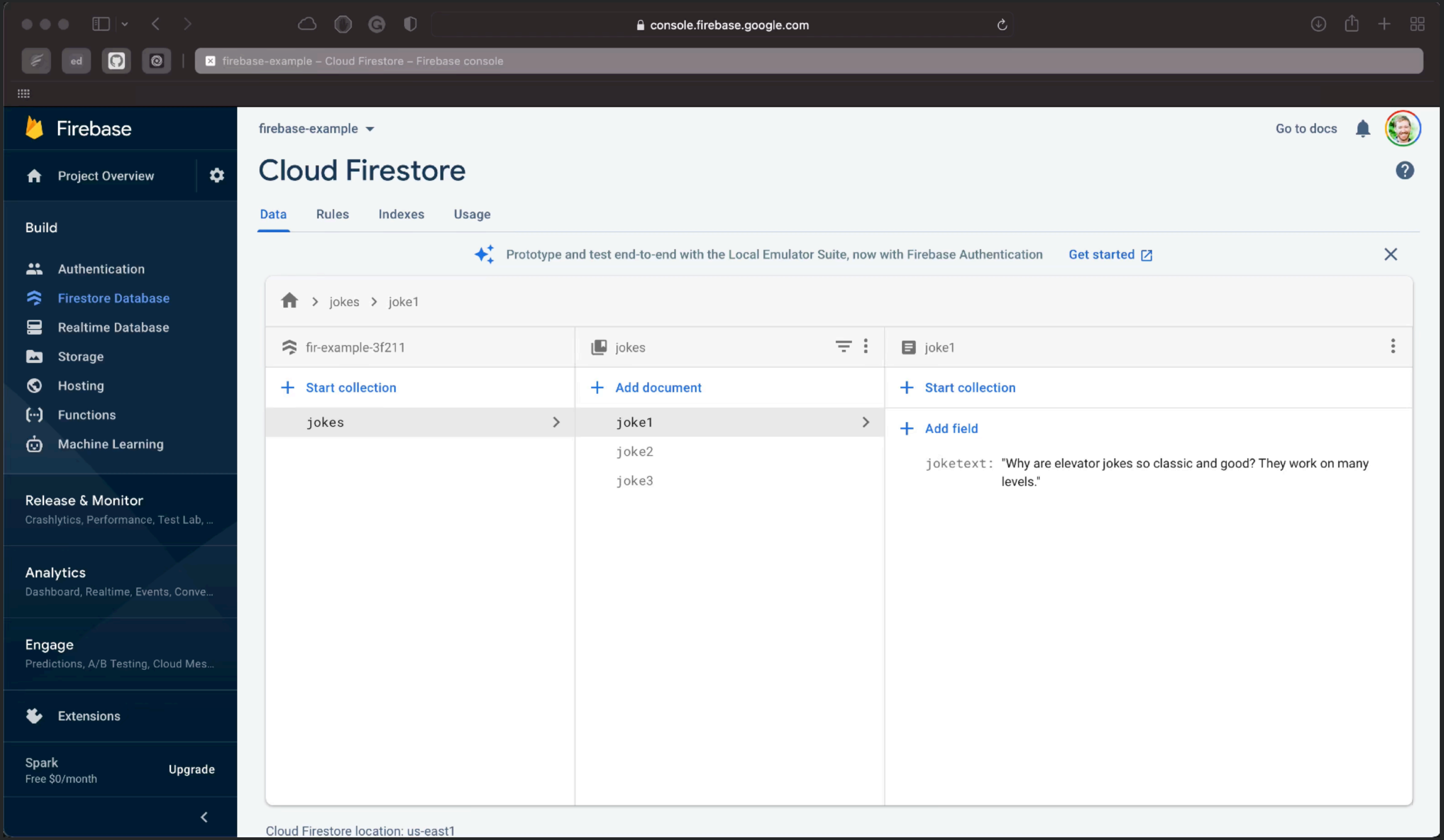


```
Legacy:Microservice-Example KevinMoran$
```

Demo: Let's build a Microservice!



Demo: Let's build a Microservice!



console.firebase.google.com

firebase-example – Cloud Firestore – Firebase console

Firebase

Project Overview

Build

- Authentication
- Firestore Database
- Realtime Database
- Storage
- Hosting
- Functions
- Machine Learning

Release & Monitor
Crashlytics, Performance, Test Lab, ...

Analytics
Dashboard, Realtime, Events, Conve...

Engage
Predictions, A/B Testing, Cloud Mes...

Extensions

Spark
Free $0/month
Upgrade

firebase-example

Go to docs

Cloud Firestore

Data | Rules | Indexes | Usage

Prototype and test end-to-end with the Local Emulator Suite, now with Firebase Authentication Get started

jokes > joke1

fir-example-3f211	jokes	joke1
+ Start collection	+ Add document	+ Start collection
jokes	joke1	+ Add field
	joke2	joketext: "Why are elevator jokes so classic and good? They work on many levels."
	joke3	

Cloud Firestore location: us-east1

In Class Activity: Modifying this MicroService + HW2

- Try implementing some new features:
  - Make the GET request return a random joke
  - Add support for different types of jokes with different fields
    - e.g. knock-knock, etc.
  - Allow for updating punchlines separate from setups
  - Use JSON request body instead of query parameters
- Feel free to work on HW2 as well!

`https://github.com/GMU-SWE432-F22/microservice-example`

Also posted on Ed Discussions

Acknowledgements

Slides adapted from Dr. Thomas LaToza's SWE 632 course