How to Deploy a Node.js API Endpoint Container on Cloud Run to CRUD Firestore Documents

COE-558: Cloud and Edge Computing
KFUPM - Term 241 - Nov 2024

Mohammad Al-Mohsin
mohammad.mohsin@kfupm.edu.sa

This guide describes how to deploy a Node.js container on **GCP Cloud Run** that performs CRUD operations on Firestore NoSQL document database.

If you already have a Firestore database and collection(s), skip to step # 3.

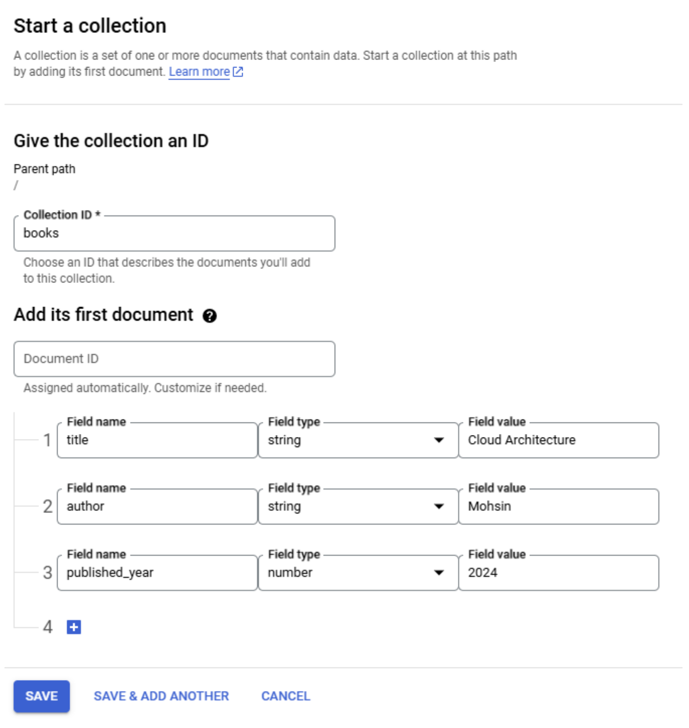
1. Go to Firestore GCP Service and Create a Firestore Database

From Google Cloud Console (https://console.cloud.google.com) navigate to Firestore. You will be notified that the Cloud Firestore API will be automatically enabled for your selected GCP project, if it is not already enabled.

1.1 Click on CREATE DATABASE button.
1.2 Select Native mode (recommended) and click CONTINUE.
1.3 Keep the default values. Make sure the Database ID is grayed out with the value (default) to qualify for the free tier. Otherwise, use a new GCP project or an existing (default) database. The **Pricing summary** on the right side should show that you are using the free tier.
1.4 Click on CREATE DATABASE.

2. Start a Collection

2.1 You should be redirected to the **Firestore Studio**at the **PANEL VIEW** automatically once the (default) database is created.
2.2 Click on + START COLLECTION.
2.3 Type a Collection ID, for example: books.
2.4 Fill in field names, types and values for a book and then click SAVE.



Start a collection

A collection is a set of one or more documents that contain data. Start a collection at this path by adding its first document. Learn more

Give the collection an ID

Parent path
/

Collection ID *: books

Choose an ID that describes the documents you'll add to this collection.

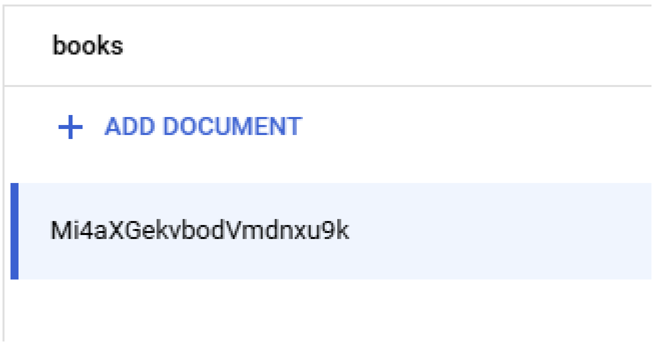
Add its first document

Document ID

Assigned automatically. Customize if needed.

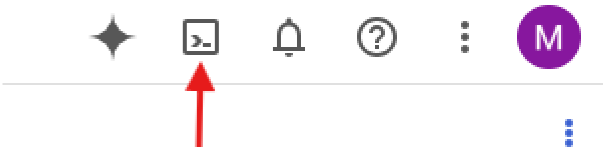
	Field name	Field type	Field value
1	title	string	Cloud Architecture
2	author	string	Mohsin
3	published_year	number	2024
4			

2.5 Add one or more books by clicking on + ADD DOCUMENT. Feel free to add more or less fields than what you created for the first book.



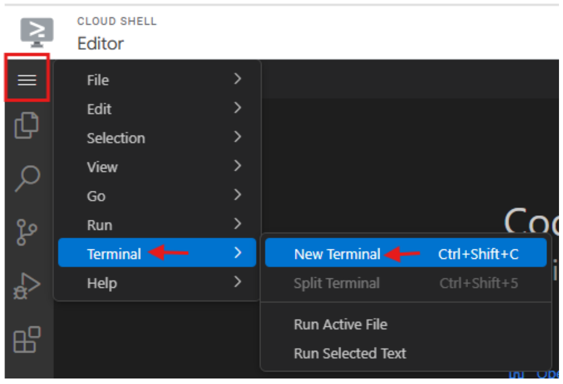
3. Node.js Application

3.1 Activate and launch Cloud Shell by following this guide: Launch Cloud Shell



3.2 Open Cloud Shell Editor by clicking on Open Editor button or navigating to https://ide.cloud.google.com/

3.3 Open a new terminal inside the Cloud Shell Editor



3.4 From the new terminal or through the Cloud Shell Editor, **create a directory** and change to it.

```
mkdir crud-firestore && cd crud-firestore
```

3.5 **Initialize Node.js** application and accept default answers. This will create package.json file for you:

```
npm init --yes
```

3.6 Install **express** web framework, **body-parser**, and Node.js Cloud **Firestore client** (SDK). They will be installed in **node_modules** directory and will be added as dependencies in **package.json** file

```
npm install express @google-cloud/firestore body-parser
```

3.7 Create index.js file with the following content

```
const express = require('express');
const bodyParser = require('body-parser');
const { Firestore } = require('@google-cloud/firestore');

const app = express();
const firestore = new Firestore();

app.use(bodyParser.json());

const PORT = process.env.PORT || 8080;

app.listen(PORT, () => {
   console.log(`Server is running on port ${PORT}`);
});

// Just for testing that our Node.js container is running fine
app.get('/', async (req, res) => {
    try {
        res.status(200).send("Hello COE 558");
    } catch (error) {
        res.status(500).send(error.message);
    }
});

// Create a new books
app.post('/api/books', async (req, res) => {
    try {
      const data = req.body;
      const docRef = await firestore.collection('books').add(data);
      res.status(201).send(`Book created with ID: ${docRef.id}`);
    } catch (error) {
      res.status(400).send(`Error creating book: ${error.message}`);
    }
 });

// Read all books
app.get('/api/books', async (req, res) => {
    try {
        const snapshot = await firestore.collection('books').get();
        if (snapshot.empty) {
            res.status(404).send({ message: 'No books found.' });
            return;
        }

        const users = [];
        snapshot.forEach(doc => {
            users.push({ id: doc.id, data: doc.data() });
        });

        res.status(200).send(users);
    } catch (error) {
        res.status(500).send(error.message);
    }
});

// Read a specific book by ID
app.get('/api/books/:id', async (req, res) => {
    try {
      const docRef = firestore.collection('books').doc(req.params.id);
      const doc = await docRef.get();
      if (!doc.exists) {
        res.status(404).send('No such book!');
      } else {
        res.status(200).send(doc.data());
      }
    } catch (error) {
      res.status(400).send(`Error reading book: ${error.message}`);
    }
});

// Update a book by ID
app.put('/api/books/:id', async (req, res) => {
    try {
```

```
    const data = req.body;
    const docRef = firestore.collection('books').doc(req.params.id);
    await docRef.update(data);
    res.status(200).send('Book updated successfully');
  } catch (error) {
    res.status(400).send(`Error updating book: ${error.message}`);
  }
});

// Delete a book by ID
app.delete('/api/books/:id', async (req, res) => {
  try {
    const docRef = firestore.collection('books').doc(req.params.id);
    await docRef.delete();
    res.status(200).send('Book deleted successfully');
  } catch (error) {
    res.status(400).send(`Error deleting book: ${error.message}`);
  }
});
```

3.8 Deploy your application to **Cloud Run** directly from the source code. This is a sample command. refer to this page for documentation and other options: gcloud run deploy

```
gcloud run deploy crud-firestore-service --source . --region=us-central1
--allow-unauthenticated
```

3.9 If you are asked to enable additional APIs and create an Artifact Registry, go ahead and enter Y

3.10 Once the container image is built and deployed to Cloud Run, you should get the service URL

```
Service [crud-firestore-service] revision [crud-firestore-service-00001-
bk8] has been deployed and is serving 100 percent of traffic.
Service URL: https://crud-firestore-service-xxxx.us-central1.run.app
```

3.11 Now you can test your API endpoint by visiting the service URL:

`https://crud-firestore-service-xxxx.us-central1.run.app/`

`https://crud-firestore-service-xxxx.us-central1.run.app/api/books`

4. Notes

4.1 If authentication is enabled on your Cloud Run service, you will need to provide the Bearer token in the Authorization HTTP header when you test for example with Postman or curl command.

```
Authorization: Bearer token
```

You can get the token for your logged-in account using this command:

```
gcloud auth print-identity-token
```

4.2 If you want to disable authentication but you couldn't due to an organization policy constraint, you can remove it using the following command. Replace xxxxx with your GCP organization ID.

```
gcloud org-policies delete constraints/iam.allowedPolicyMemberDomains --
organization=xxxxx
```