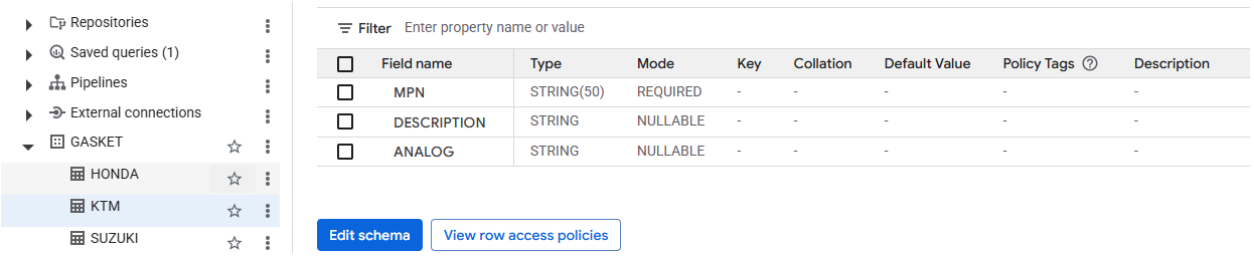
I've always loved Serverless technology - it feels a bit magical. We send a request to a site that doesn't even exist yet, but it comes to life and responds once the request is made.

During free time after my last very specific work I decided to start my microbusiness (hobby) where I design different types of gaskets for auto- moto- equipment. And one day I understood I was drowning in different folder names, part numbers, photos etc…

So I decided - it needs to catalog and make a structure on my website. Solution will be rollouted on the GCP Cloud
How I choose DB. As database I chose Google BigQuery as data store which do not need run as dedicated service, it already present in Cloud support SQL query language and have a very low price* for my needs

* Big Query - The first 1TiB requested data per month is free.

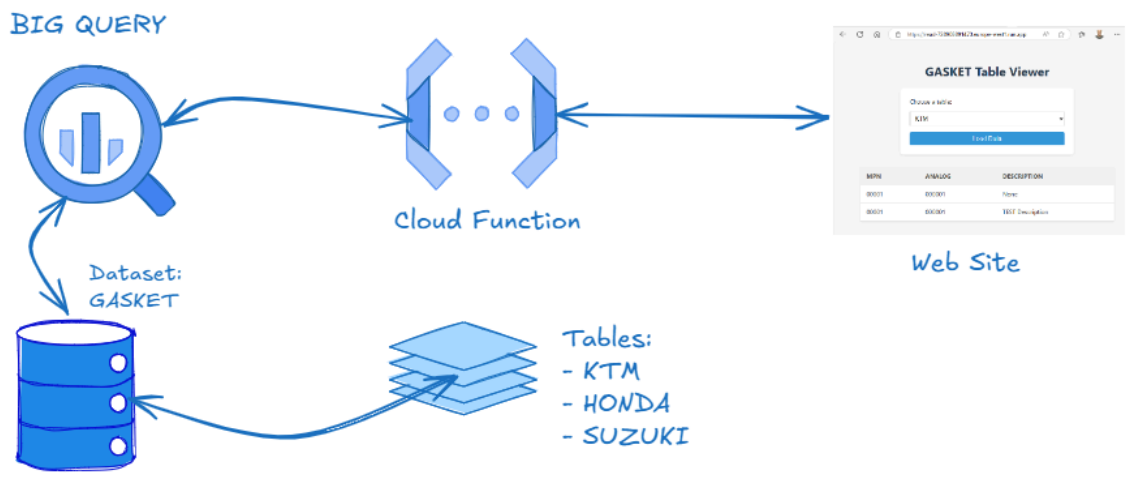
I created a Dataset GASKET with several tables inside KTM, HONDA Suzuki. Each table has a field MPN ANALOG DESCRIPTION. It ~~not~~ enough for me 🙂



Lets go step-by-step

1. Create in BQ new Dataset with the name GASKET. No comment anymore.
2. Inside Dataset create a new Table with name KTM with 3 STRING fields.
3. Let bootstrap data to Table **KTM** in BigQuery Dataset **GASKET.** Upload test data inside Table KTM using SQL Query UI :

```
INSERT INTO `<PROJECT_ID>.GASKET.KTM` (MPN, ANALOG, DESCRIPTION)
VALUES ('00001', '10000', "TEST Description");
```



4. Now I need to make a serverless solution which will be connected to BigQuery read and return data in my web site in table view - Serverless Engine. I stopped at Google Cloud Function which supports Python language as a solution for doing anything in underhood mode - compile a new docker image after code update, update revision, store your revision code in a bucket and all of them in automatic mode.
   Goto CloudFunction Menu and select Use an inline editor to create a function

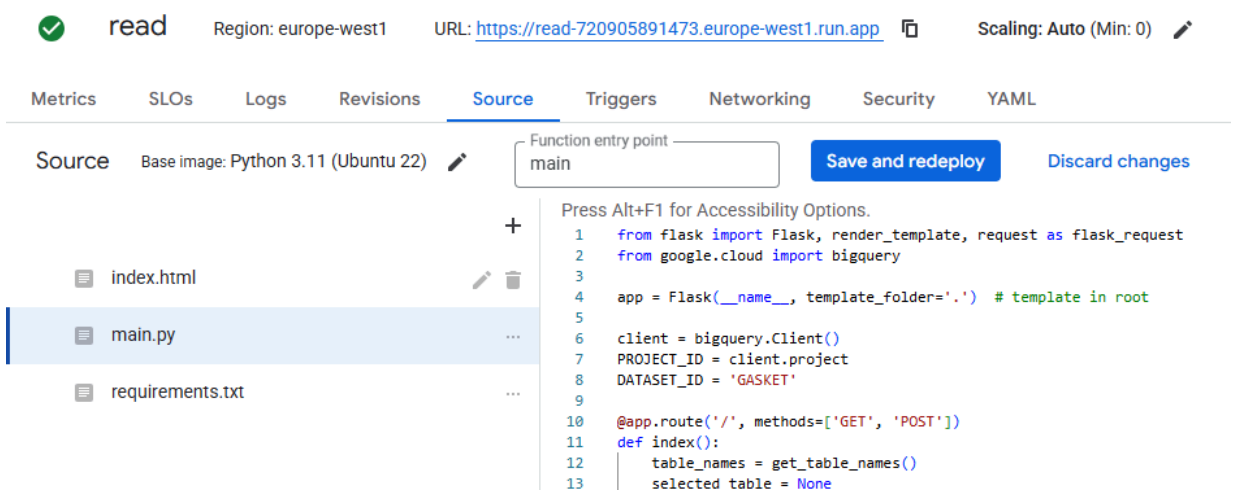
https://console.cloud.google.com/run/create?enableapi=false&deploymentType=function&hl=en&inv=1

Fill couple field in CloudFunction configuration:

- Define Service name - our CloudFunction Name
- Allow to make unauthenticated requests
  - ☒ Allow unauthenticated invocations

And click on **Create**

After that, in Source Tab creating 3 files



main.py (our logic)

```
from flask import Flask, render_template, request as flask_request
from google.cloud import bigquery

app = Flask(__name__, template_folder='.')  # template folder in root

client = bigquery.Client()
PROJECT_ID = client.project
DATASET_ID = 'GASKET'

@app.route('/', methods=['GET', 'POST'])
def index():
    table_names = get_table_names()
    selected_table = None
    records = []

    if flask_request.method == 'POST':
        selected_table = flask_request.form.get('table')
```

```
        if selected_table:
            records = get_mpn_analog(selected_table)

    return render_template('index.html', tables=table_names, selected=selected_table, records=records)

def get_table_names():
    tables = client.list_tables(f"{PROJECT_ID}.{DATASET_ID}")
    return [t.table_id for t in tables]

def get_mpn_analog(table_name):
    query = f"""
        SELECT MPN, ANALOG, DESCRIPTION
        FROM `{PROJECT_ID}.{DATASET_ID}.{table_name}`
        LIMIT 100
    """
    query_job = client.query(query)
    return list(query_job.result())

# ✅ Entry point for Google Cloud Functions (no functions-framework needed)
def main(request):
    return app(request.environ, start_response=lambda status, headers: None)
```

requirements.txt (python components, libs)

```
Flask==2.3.3
google-cloud-bigquery==3.20.0
pyarrow>= 3.0.0
```

Index.html (our web UI - yes sound good, looking bad)

```
<!DOCTYPE html>
<html>
<head>
    <title>GASKET Viewer</title>
</head>
```

```
<body>
    <h1>Select Table from GASKET Dataset</h1>
    <form method="POST">
        <label for="table">Choose table:</label>
        <select name="table" id="table">
            {% for table in tables %}
                <option value="{{ table }}" {% if table == selected %}selected{% endif %}>{{ table }}</option>
            {% endfor %}
        </select>
        <button type="submit">View</button>
    </form>
    {% if records %}
        <h2>Records from {{ selected }}</h2>
        <table border="1">
            <tr><th>MPN</th><th>ANALOG</th><th>DESCRIPTION</th></tr>
            {% for row in records %}
                <tr>
                    <td>{{ row.MPN }}</td>
                    <td>{{ row.ANALOG }}</td>
                    <td>{{ row.DESCRIPTION }}</td>
                </tr>
            {% endfor %}
        </table>
    {% endif %}
</body>
</html>
```

Don't forget to define in CloudFunction Function entry point **main** and click **Save and Deploy**.

For running a serverless solution there is enough 256MB RAM and 1vCPU. Cold start gets 2-3second. And the main thing in my opinion is the light weight container - only 65MB image size. So Win-Win.

This is very simple case show us how easy build intercommunication between different Services in GCP make a Simple homepage and don't care about server electricity internet connection and pay nothing for this 😂

Link on GitHub code: [dnk80/Serverless_Computing: Serverless Computing] Link on site [https://read-720905891473.europe-west1.run.app]